Rapport soumis aux rapporteurs, dans le but de sanctionner le
dossier pour l'obtention du grade de
Docteur en Informatique
d'Aix*Marseille Université

# From Confusion Noise to Active Learning: Playing on Label Availability in Linear Classification Problems

Ugo Louche, le 13 Mai 2016

Mots-clés : modèles linéaires, classification, multi-classe, matrice de confusion, bruit, apprentissage actif, géométrie computationelle, perceptrons, méthodes de plans coupants, schémas de compréssion

# Contents

# List of Figures

# List of Algorithms

# Introduction

Machine learning originates from Artificial Intelligence's quest to reproduce what is arguably one of the core features of an intelligent behavior: the ability to be taught. That is, more than the ability to repeat mindlessly a lesson, to learn complex concepts from limited experience and examples alone. In doing so, Machine Learning rapidly outgrew the field of Artificial Intelligence, which was focused on other, wider, problems, to become a scientific field of its own. The (short) history of Machine Learning is exhilarating and a topic worth discussing in itself although here is not the place to recount it. Suffices to say that in the absence of any formal definition of human learning, Machine Learning established itself at the crossroad between a diverse scientific fields interested in uncovering and generalizing patterns from data. Throughout the years, Machine Learning has never ceased to be fueled by theory and results from many scientific fields from Mathematics and Computer Science. As such, modern Machine Learning is a particularly heterogeneous topic with strong influences from multiple fields such as but not limited to Computational Statistics, Signal Processing, Language Processing, Neuro-Computing, Computer Vision, Convex and non-Convex Optimization, Linear Programming, Data Mining, Information Retrieval and of course Theoretical Computer Science through Computational Learning Theory.

More concretely, Machine Learning is interested in devising systems that are able after observing a particular behaviour in various situation to accurately mimic said behaviour on a previously unseen environment. If we refer to the environment as input and the behaviour as output, the task of learning is thus to correctly predict the outputs associated with some new inputs based on previous observations of input/output pairs and we refer to this particular setting as *supervised learning*.

One of the major topics Machine Learning is interested in is the one of Concept learning, that is to separate inputs that match a given concept from the others. A textbook example of concept learning that has found its way into our daily lives is *SPAM* automatic detection, where in this case the inputs are the various emails one receives and the outputs consist of the values $+1$ and $-1$ whether the message is a *SPAM*. Hence, learning to distinguish between *SPAM* and non-*SPAM* emails amounts to learn the concept of what a *SPAM* is. More generally, concept learning or, alternatively, classification, refers to the problem of predicting whether an input belongs to a concept based on previous observations. A variation of the problem of classification that we may mention is Multiclass Classification, where instead of predicting membership to some specific concept, the learners task is to discriminate the inputs into multiple categories. A practical example of Multiclass Classification is the problem of handwritten digits recognition where the classification task is that given a picture of a handwritten digit to be able to determine which digit it is; a task for which modern automatic recognition algorithm are on par with human capabilities [Ciresan et al., 2012].

Classification will be the central focus of this thesis, whether it is multiclass or binary classification. Formally, a working assumption in classification problems is that the data, that is the inputs, are vectors in a Euclidean space of arbitrary dimension and the outputs are either $+1/-1$ for binary classification or an integer between 1 and $Q$ in the case of multiclass classification with $Q$ classes. The usual setting in Machine Learning is the so-called *supervised setting* where it is assumed the existence of a *training set* $\mathcal{S}$ composed of input/output pairs, where the input is thus a vector and the output a scalar value. We formally call target concept and write $t$ the concept we wish to learn, that is $t$ is a mathematical function that maps each input vector with its associated class. The task of learning in itself consist in inferring the correct classification rule from $\mathcal{S}$ only, that is to find a function $h$ that will mimic $t$ as closely as possible. Supervised learning has long been studied and represents what is usually accepted as the most common setting for machine learning as it is both a simple and intuitive paradigm but also allows for interesting and insightful theoretical results.

## Motivations

Let us motivate our work through an example and consider the (fictive) problem of categorizing different pictures of vertebrate animals with respect to their subcategories, that is fishes, amphibians, reptiles, birds and mammals; notably, this problem is inspired from the iconic Animal With Attribute learning task [Lampert et al., 2009]. As it is, constituting a supervised dataset for this problem seems easy enough, each of the five groups have easily identifiable visual cues that most people learned to recognize. For instance, fishes have fins, reptiles have scales, mammals have furs or hair, birds have feathers and amphibian fit neither of those previous rules and their skin is naked. Hence based on those simple rules it seems that pretty much anybody with an internet access can gather a dataset for problem and label it at minimal costs.

The point is these rules are simplistic and although they may work in general, some cases may require more thought before deciding which category they belong. For instance, whales may seem closer to fishes than mammals, moreover there is sometime a thin line between whales and sharks which are from two different groups, which is something that may be confusing. Additionally, categorizing some species in any group at all may prove challenging for anyone without an expert knowledge. Such is the case of the platypus and although it is now known that the platypus belongs to the group of mammals (despite its beak, venomous fangs and the fact that it lays eggs) the discovery of the specie by European travellers originally sparked controversy even among experts and the animal was for some time considered a hoax. Additionally, far less iconic species may also prove challenging —see e.g. the Axolotl also known as "Walking Fish"— and more than an peculiar oddity these difficult cases represent a reality that should be accounted for.

From a higher point of view, we may ask the question of the relevance of those underrepresented species with respect to the problem as a whole. The fact is, among the nearly $70,000$ vertebrates species referenced,

half of them are fishes and less than 10 percents are mammals; moreover, vertebrates only amounts for approximately 5 percents of the described animal species in general. Arguably, a classifier that predicts every vertebrates to be fishes is no better than one that does the same with mammals, however, assuming that each species are equally represented in the pictures, the former will have a 50% accuracy rate (that is, half the examples will be correctly classified) but the latter will only be correct on 8% of the pictures. Put otherwise, concept learning is about accurately learning concepts rather than making as few mistakes as possible and the same applies to those limit cases which are difficult to categorize: they are part of the concepts we are trying to learn, independently of how underrepresented they are with respect to the problem as a whole.

The message behind this example is that more often than not, accurately labelling the data is difficult and there is actually a great disparity in label and data availability. In the example above, this comes from the fact that, for some data, an expert is needed in order to properly categorize an image, this is both a slow and a costly process as opposed to acquiring the data which can be automated without much hassle. This disparity is often unaccounted for in the supervised setting and as a result, one must either deal with mislabeled data or simply ignore a part of the data for which reliable labels cannot be obtained.

At this point we may also mention the semi-supervised setting which aims to answer this problem by taking into account both labeled and unlabeled data. Notably semi-supervised learning has been a vivid topic in machine learning for the past years [Zhu, 2005]. Ultimately, semi-supervised learning is more of a passive solution to the problem of label scarcity whereas the present work seeks to explore proactive approaches for coping with the inherent difficulty of obtaining labels and although semi-supervised learning is closely related to some of our contribution it is a setting that eventually falls outside of the scope of this thesis.

The main idea driving this thesis is that label availability should not be considered as a hard limit to the constitution of a training set but more of an adjustment variable that one can tweak and play with. More precisely, our message is that one can play with label (un)availability by willingly allowing for some labelling errors or, at the contrary, asks for fewer, informative, labels.

The first approach leads to a supervised learning problem where some labels are wrong, keeping our previous example in mind this correspond to the case where the pictures are not labelled by an expert and as such some errors are made, however chances are that those errors will follow some kind of pattern. For instance, in regard to the whale/shark discussion above, we can expect some errors between fishes and mammals, moreover Axolotl and other similar species are expected to sometimes be categorized as fish rather than amphibians. On the other hand, fishes and birds are two distinctive groups and very few mistakes should be expected between those two groups. As for the second approach, a solution is to allow for interactivity between the learner and the expert in order to reduces the number of labels required for learning. In a nutshell, the idea is to let the learning algorithm actively decides which data are relevant to its learning procedure rather than providing it with an already

labelled dataset. Through this interactivity we aim to reduce the number of labelled pictures needed and only focus on the interesting ones, that is the ones we truly need an expert for. Typically, in the example discussed so far, it is expected that a few queries are made to roughly identify the five groups then most of the subsequent label requests would be about difficult cases where the categorization problem is not trivial.

## A more formal approach

In addition to the above example, we shall formally lay out the foci of this thesis as well the contribution we propose. As stated before, from a practical standpoint this work revolves around two settings that we may motivate as solutions to overcome the issues of label unavailability classification problems. Moreover, a theoretical and transverse theme of the present work is to discuss the problem of classification through the various fields related to Machine Learning, as a result the domain of relevance of our contributions ranges from statistical learning theory to computational geometry.

### Confusion Problems

Dealing with errors in the training set is arguably as old as classification itself, the fact is whether it is from some corrupting processes or a lack of proper access to the target concept, machine learning has to cope with dataset that might contains erroneous labels. Unfortunately strong negative theoretical results were given early on for learning under the presence of labeling mistakes in the general case (see e.g. [Höffgen et al., 1995]) which motivated a shift in the literature to slightly more specific settings. The one we are interested in is that of *confusion noise* when the labelling errors in the dataset only depend on the classes and not the location of the data. In a binary setting it means that the errors are characterized by their mislabeling probability with respect to the positive and negative classes alone and past works have shown that learning can be achieved in binary classification under the presence of confusion noise [Blum et al., 1998, Bylander, 1994]. We will investigate how this work can be extended to the problem of multiclass classification to which confusion noise naturally extend, notably the noise will be tied to a *confusion matrix* containing the mislabelling probability rates between each possible pairs of classes. One of our contribution is to propose a novel algorithm to deal with the multiclass confusion setting which we call UMA. We provide a theoretical analysis of UMA and demonstrate its statistical correctness with respect to the problem of confused multiclass learning. Moreover, UMA is, to the best of our knowledge, the first algorithm to be provably efficient in this setting.

### Active Learning

Active learning is a natural and elegant way to overcome the label requirement of the supervised setting. From a theoretical perspective, supervised learning requires the training set to be big enough to ensure that the only

consistent solution with the labeled data is sufficiently close to the target concept. Active learning, on the other hand, iteratively build a solution from a few labeled examples and interactively asks for the most relevant labels retrospectively to the current set of consistent hypotheses. The end result is an algorithmic scheme that starts from an unlabeled dataset and iteratively asks for more labels. Thus, active learning algorithms generally may require far less labels than their supervised counterpart at equal performance rates. Recent advances in active learning pointed at a possible geometric interpretation of the problem that seemed promising but ultimately was left unexplored [Tong and Koller, 2001], notably the Active Learning algorithm proposed was functionally close to localization methods such as Cutting Planes algorithms. We will propose a new interpretation of active learning built upon these geometrical considerations and more particularly a new family of active learning algorithms based on this geometric interpretation. Additionally this family generalizes some well-known, state-of-the-art, active algorithms and our geometrical interpretation will shed new lights on the capacities and limitations of those. We will therefore propose a variation of the algorithm proposed in [Tong and Koller, 2001] that is both theoretically justified and validate our analysis through experimental results.

### Theoretical aspects

Learning in general, and particularly classification, is known to be strongly related to the problem of solving a set of linear equations. An interpretation of learning is thus to solve a linear programming problem, a link that is well known in the machine learning community but surprisingly rarely considered. Linear programming methods are usually used as off-the-shelf tools and their features from a Machine Learning standpoint tend to be overlooked. In this thesis, we will provide a detailed review of how classification and linear optimization precisely relates to each other. Moreover, we will focus on a geometric interpretation of classification where the task of learning amounts to localizing a vector into some convex version space. Building on these ideas and the peculiar needs of machine learning, we will not only propose a new compound algorithm that combines the strengths of both machine learning's perceptron and Cutting Planes methods but also give a new theorem on the partitioning properties of approximate centers of gravity. Notably, this last contribution is not limited to the setting of machine learning and is relevant to all methods relying on the partitioning properties of centers of gravity.

## Outlines

This thesis is organized along three parts. In an attempt to propose, as much as possible, a self-contained content the first part is dedicated to the formal introduction of the various notions and ideas behind linear classification that we will use throughout this work. Because providing an extensive review of even the most fundamental notions of linear classification is unrealistic, this part should not be taken as a work of review but merely as a warm-up for the remaining of this thesis and a mean to

gently introduce our notations and formalisms. Chapter 1 will deal with the most fundamental matters while chapter 2 will provide an additional layer of refinement and extend the ideas presented previously. By the end of part I most of the non-specific theoretical ground for this thesis will be introduced and the two remaining parts will focus on the details of our contributions and thus are independent from each other.

Namely, part II tackles the problem on confusion learning and chapter 3 is devoted to a review of the state-of-the-art for bi-class confused learning as well as the introduction of confusions matrices as both an error measure and a noise descriptor. Chapter 4 will present our first contribution, a study of the problem of confused learning in a multiclass setting, which was first published at the *Asian Conference on Machine Learning* in 2013 [Louche and Ralaivola, 2013] and as an extended version in *Machine Learning Journal* [Louche and Ralaivola, 2015b].

The last and third part of this thesis is three-fold an while the motivation throughout this part is the application of Cutting Planes methods to Active Learning algorithms, each chapter has its own focus and can be taken independently. In chapter 5, we will take a step sideways and consider the problem of localizing a point within the confines of a version space, a problem that is strongly related to classification. In particular, the focus of this chapter will be on geometrical aspect and Cutting Planes methods. Moreover, one of our contribution in chapter 5 is theorem 5.2 which is an extension of the fundamental theorem of centers of gravity (theorem 5.1 [Grunbaum, 1960, Levin, 1965]) to approximate centroids. Chapter 6 deals with general machine learning in light of the discussions of the previous chapter and will notably discuss how Cutting Planes methods are naturally fit for machine learning problems, something we demonstrate by proposing a new hybrid algorithm that theoretically and practically combines the properties of a Perceptron algorithm with the update scheme of Cutting Planes methods. Lastly, chapter 7 will introduce the setting of Active Learning and discuss how the general active learning process is similar to the Cutting Planes update scheme. Additionally, in regard of the previous discussions in chapters 5 and 6 we will provide new theoretical insights on the work of [Tong and Koller, 2001] and propose a theoretically sound framework as well as an improvement of their algorithm based on our theoretical analysis. Finally, note that the contributions of these las three chapters were all first published together at the *International Joint Conference on Neural Network* in 2015 [Louche and Ralaivola, 2015a] where it won the *best student paper* award.

# Notations

**Mathematical Notations**

| | |
|---|---|
| $\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}, \boldsymbol{v}, \ldots$ | Vectors |
| $\mathbf{W}, \mathbf{M}, \mathbf{C}, \ldots$ | Matrices |
| $\mathbf{W}_{\cdot i}$ | The $i$th column of the matrix $\mathbf{W}$ |
| $\mathbf{W}_{i\cdot}$ | The $i$th row of the matrix $\mathbf{W}$ |
| $\mathbf{W}^\top, \boldsymbol{x}^\top$ | The Matrix and vector transposes |
| $\|\cdot\|_2$ | The Euclidian vector norm |
| $\|\cdot\|_\mathrm{F}$ | The Frobenius matrix norm |
| $\langle \cdot ; \cdot \rangle$ | The inner product of two vectors |
| $X, T, \ldots$ | Random Variables |
| $\mathscr{D}, \mathscr{D}_t, \mathscr{U}, \ldots$ | Distributions |
| $\mathbb{E}_{\mathscr{D}}[\cdot]$ | The expectation of a random variable with respect to the distribution $\mathscr{D}$ |
| $\mathbb{P}_{\mathscr{D}}(\cdot)$ | The probability of an event with respect to the distribution $\mathscr{D}$ |
| $\mathcal{C}, \mathcal{W}, \mathcal{E}, \ldots$ | Sets |
| $\|\cdot\|$ | The size of a set / the absolute value of a scalar |
| $[N]$ | The set of integers $\{1; \ldots; N\}$ |
| $\mathcal{O}(\cdot)$ | The Big O notation |

**Commonly Used Terms**

| | |
|---|---|
| $\mathbb{H}, \mathbb{X}, \mathbb{Y}$ | Hypothesis, Input and Output spaces |
| $\boldsymbol{x}$ | A learning example |
| $t(\cdot)$ | The target concept |
| $h_{\boldsymbol{w}}(\cdot)$ | The linear classifier of normal vector $\boldsymbol{w}$ |
| $\gamma_{\boldsymbol{x}}^{h_w}, \gamma_{\mathcal{S}}^{h_w}$ | The margin of $h_{\boldsymbol{w}}$ over a training data $\boldsymbol{x}$ and across a training set $\mathcal{S}$ |
| $\mathscr{D}$ | The unknown data distribution over $\mathbb{X}$ |
| $\mathscr{D}_t$ | The joint data distribution over $\mathbb{X} \times \mathbb{Y}$ consistent with $t$ |
| $\mathcal{S} \overset{\text{i.i.d}}{\sim} \mathscr{D}_t^N$ | The training set $\mathcal{S}$ composed of $N$ independent realizations of the joint distribution $\mathscr{D}_t$ |
| $l_{0-1}(\cdot, \cdot)$ | The 0-1 loss |
| $R_{\mathscr{D}_t}^{l_{0-1}}(\cdot)$ | The 0-1 risk of a classifier with respect to the joint distribution $\mathscr{D}_t$ |
| $R_{\mathcal{S}}^{l_{0-1}}(\cdot)$ | The empirical 0-1 risk of a classifier with respect to $\mathcal{S}$ |
| $Q$ | The number of classes in multiclass problems |
| $\mathbf{C}$ | The confusion matrix of a noising process |
| $\mathbf{0}_D$ | The zero vector of dimension $D$ |
| $\mathbf{I}_D$ | The identity matrix of dimension $D$ |
| $\mathbb{I}[\cdot]$ | The indicator function |
| $\mathcal{B}_1$ | The Ball of unitary Euclidian norm |
| $\mathcal{C}$ | The search space |
| $\mathcal{W}$ | The version space |
| $\mathrm{Vol}(\cdot)$ | The volume of a convex body |
| $CC(\cdot), CG(\cdot)$ | The Chebyshev and gravity centers of a convex body |
| $O(\cdot)$ | An Oracle function |

# Part I

# Preliminaries

# The Problem of Classification

1

Contents



The present chapter aims at providing a gentle start to the ideas of linear classification. While we do not expect the topics exposed here to be new for most readers, this chapter should be seen as a mean to introduce our notation and underline the peculiarities of our setting. Notably we give a hyperplane's definition (definition 1.1) that is rather unusual in the machine learning literature. Although our definition is equivalent to the usual one, it will allow for an easier exposition of our contributions throughout this thesis.

More generally, this chapter serves the purpose of giving a rough direction to this thesis by laying out its fundamental ground. Hence, this chapter is heavily biased toward the problems we will tackle later on in Part II and III, that is noisy multiclass classification and active learning. As such, this chapter is in no way an extensive exposition of the foundations of machine learning and more often than not we will refer the reader to external references as soon as our discussion leaves the boundaries of what is needed for this thesis to be understandable. To some extent, that last remark holds for the various definitions and theorems to come in the sense that we will usually favor a simpler formulation over an unneeded, more general statement.

## 1.1 The scope of this Thesis

As in everything, we first have to clarify, formally, what belongs within the boundaries of this work and what does not, if we ever want to move on more complex matters. This section's aspiration is simply to precisely draw the line between what is relevant to this thesis and what is not. In other words, before anything else we have to discuss, even briefly, what are these mathematical objects we are interested in.

### 1.1.1 A story of spaces and problems

Let us start with the *input space*, $\mathbb{X}$, where our data lives. Throughout we will assume $\mathbb{X}$ to be a Euclidean space of predetermined dimension $D$. Therefore, data are seen as vectors in $\mathbb{X}$ of dimensions $D$. In practice however, data are usually real-valued vectors and most of the time it is fair to assume that $\mathbb{X}$ is akin to $\mathbb{R}^D$.

Associated with $\mathbb{X}$ is the *output space* $\mathbb{Y}$ which is the space of values that will be matched with each data. With different $\mathbb{Y}$ comes different machine learning problems. Regression problems correspond to the case where $\mathbb{Y}$ is a continuous space, while classification problems imply a discrete and finite $\mathbb{Y}$. This thesis will focus on classification problems and as such regression will be left mostly untouched hereafter. More precisely, we will be interested in the so-called bi-class case and its multiclass extension, that is, respectively: $\mathbb{Y} \doteq \{-1; 1\}$ and $\mathbb{Y} \doteq \{1, \dots, Q\}$ where $Q \in \mathbb{N}$ is the number of classes in the multiclass setting.

However, as far as this chapter is concerned we will only discuss the problem of binary classification, hence, unless specified otherwise,

$$\mathbb{Y} \doteq \{-1; 1\}$$

This will save us a lot of hassle and allow for streamlined notations while we will be expounding the basic notions of classification. Multiclass classification will then be properly reintroduced in Section 2.3.

### 1.1.2 Classes and hypothesis

Put simply: machine learning is about inferring some functions from a training sample that map $\mathbb{X}$ to $\mathbb{Y}$ and while the hows will be made clear in a while, we will first introduce some notions.

We distinguish two semantic groups amongst those mappings referred to as *concepts* and *hypotheses*. Both are set of functions from $\mathbb{X}$ to $\mathbb{Y}$ and the distinction between the two is only in how they relate to the task of classification. Namely: *concepts* are arbitrary and unique to each problem, they are the function to be inferred and as such are generally unknown to the user; conversely, the *hypotheses* are candidate functions able to mimic a given *concept* and the pith of learning is to find the one hypothesis that is the most similar to that *concept*.

Hypotheses are bound to a *hypothesis class* or, interchangeably, *hypothesis space* noted $\mathbb{H} \subset \mathbb{Y}^{\mathbb{X}}$, where $\mathbb{Y}^{\mathbb{X}}$ denotes the set all functions from $\mathbb{X}$ to $\mathbb{Y}$. The concept —noted $t$ hereafter— on the other hand, may or may not be an element $\mathbb{H}$ and different problems arise whether $t \in \mathbb{H}$; yet, it is

unnecessary and cumbersome for now to delve into these considerations without a proper definition of learning.

Many different hypothesis class have been studied in machine learning. Among them is the class of *linear classifiers* which is arguably one of the most popular. A tremendous amount a research has focused on this particular class, leading to some of the most well-known methods of machine learning (e.g. [Boser et al., 1992]).

In a nutshell, linear classifiers are hyperplanes in $\mathbb{X}$ (that is usually $\mathbb{R}^D$). Thus they split $\mathbb{X}$ in two parts, inducing a mapping from $\mathbb{X}$ to $\mathbb{Y}$. More formally

**Definition 1.1** (Linear Classifier) *We call* linear classifier *of normal vector* $\boldsymbol{w} \in \mathbb{X}$*, the function* $h_{\boldsymbol{w}} \in \mathbb{Y}^{\mathbb{X}}$ *such that for any point* $\boldsymbol{x} \in \mathbb{X}$*:*

$$h_{\boldsymbol{w}}(\boldsymbol{x}) \doteq sign\left[\langle \boldsymbol{w}; \boldsymbol{x} \rangle\right]$$

*where* $sign(0)$ *is arbitrarily set to* $+1$ *and* $\langle \cdot; \cdot \rangle$ *denotes the inner product in* $\mathbb{X}$*.*

Alternatively, any $\boldsymbol{w} \in \mathbb{X}$ implicitly defines a *linear classifier*:

$$\boldsymbol{w} \equiv h_{\boldsymbol{w}}$$

and both $h_{\boldsymbol{w}}$ and $\boldsymbol{w}$ will be used interchangeably. In other words, since this thesis will exclusively focus on linear classifiers, and because linear classifiers are akin to a vector in $\mathbb{X}$, unless specified otherwise, we will now assume that:

$$\mathbb{H} \equiv \mathbb{X}$$

It should be noted that this definition does not match the one usually found in machine learning textbooks. Instead, hyperplanes as usually defined with respect to a normal vector and an offset term. The fundamental difference being that in our case hyperplanes *must* pass through $\mathbb{X}$'s origin. However, the two definitions can be seen as equivalent and one can seamlessly transition from one definition to another at the cost of an additional dimension by embedding the offset term into the hyperplane's normal vector (see, e.g., [Dunagan and Vempala, 2008] for more details).

To conclude this section, it should be noted that the semantic behind the various notions we have introduced is of utmost importance, especially in the context of the present memoir where hypotheses and data share a common representation, that is, a vector in $\mathbb{X}$. Additionally, this may also be beneficial, and switching between representations may yield unsuspected results. In particular, this is one of the premises upon which Part III is built.

## 1.2 Risks and Losses

### 1.2.1 Losses

Now that we have properly defined the hypothesis space, we may introduce the fundamental tools we will use to measure the discrepancy between hypotheses and concepts. Namely, we call *loss* some functional

$$l : \mathbb{H} \times \mathbb{X} \times \mathbb{Y} \to \mathbb{R}_+$$

Conceptually, losses are a mean to measure disagreement between two functions in $\mathbb{Y}^{\mathbb{X}}$. Different losses imply different disagreement measures and each have their pros and cons. The most intuitive —but also the less practical— loss is the 0-1-loss which is defined as follow:

**Definition 1.2** (0-1-loss) *Given a hypothesis $h \in \mathbb{H}$, a concept $t$ and a datapoint $\boldsymbol{x}$, the* 0-1-loss *of $h$ with respect to $t$ on $\boldsymbol{x}$ is defined as:*

$$l_{0-1}(h, \boldsymbol{x}, t(\boldsymbol{x})) \doteq \begin{cases} 0 & \text{if } h(\boldsymbol{x}) = t(\boldsymbol{x}) \\ 1 & \text{else} \end{cases}$$

Although the 0-1-loss is easy to fathom, it is otherwise hard to work with as the function is not continue on $\mathbb{H}$ (figure 1.2). Through the years, machine learners have devised surrogate losses to palliate the difficulties of working directly with the 0-1-loss. For linear classifiers, those surrogates typically rests on the notion of margin and are defined as a continuous and derivable function of the margin (see also figure 1.1).

**Definition 1.3** (Margin of a Point) *Let set $\boldsymbol{x} \in \mathbb{X}$ and $h_{\boldsymbol{w}} \in \mathbb{H}$ such that $h_{\boldsymbol{w}}$ is of normal vector $\boldsymbol{w}$. We define the margin $\gamma_{\boldsymbol{x}}^{h_{\boldsymbol{w}}}$ of $h_{\boldsymbol{w}}$ on $\boldsymbol{x}$ as:*

$$\gamma_{\boldsymbol{x}}^{h_{\boldsymbol{w}}} \doteq \langle \boldsymbol{w}; \boldsymbol{x} \rangle$$

Some examples of loss functions defined that way include the *quadratic loss*: $l_{\text{quad}}(h_{\boldsymbol{w}}, \boldsymbol{x}, t(\boldsymbol{x})) \doteq (t(\boldsymbol{x}) - \gamma_{\boldsymbol{x}}^{h_{\boldsymbol{w}}})^2$ and the *exponential loss*: $l_{\exp}(h_{\boldsymbol{w}}, \boldsymbol{x}, t(\boldsymbol{x})) \doteq \exp(-t(\boldsymbol{x})\gamma_{\boldsymbol{x}}^{h_{\boldsymbol{w}}})$. Finally, we introduce the *hinge loss* which is one of the most popular losses in linear classification and, as such, will be of particular interest for the following of this thesis:

$$l_{\text{hinge}}(h_{\boldsymbol{w}}, \boldsymbol{x}, t(\boldsymbol{x})) \doteq \max\{0, 1 - t(\boldsymbol{x})\gamma_{\boldsymbol{x}}^{h_{\boldsymbol{w}}}\}$$

### 1.2.2 Risks and Training Set

#### Training set

Until now, we have left unanswered the details of what we usually refer as *learning* and stood by the idea that, somehow, the task consist in finding a hypothesis $h \in \mathbb{H}$ that mimics some concept $t$. The catch here is that we cannot manipulate $\mathbb{X}$ as a whole because it likely contains an infinite number of elements and no representation of finite size. Hence the idea is to sample $\mathbb{X}$ and work with a finite sample of the set. A classical assumption that we retain throughout is that this sampling is done with respect to a fixed, yet unknown, distribution $\mathscr{D}$ and each sample is identically and independently distributed (i.i.d for short) according to $\mathscr{D}$. The intuition behind $\mathscr{D}$ is that all element in $\mathbb{X}$ are not all observed with the same frequency, particularly, $\mathbb{X}$ might contains elements that are very unlikely to be observed through the sampling procedure. Additionally, we assume the samples to be drawn simultaneously with their associated classes and define the notion of *training set* $\mathcal{S}$:

$$\mathcal{S} \doteq \{\boldsymbol{x}_i; t_i\}_{i \in [N]}$$

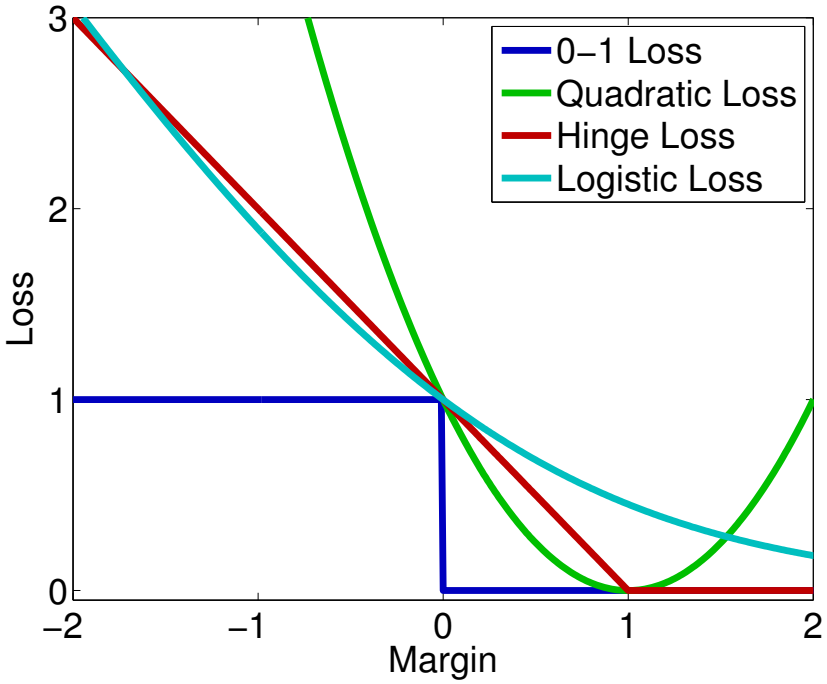



Figure 1.1 – *The 0-1, quadratic, hinge and logistic losses with respect to the classifier margin for a given point.*

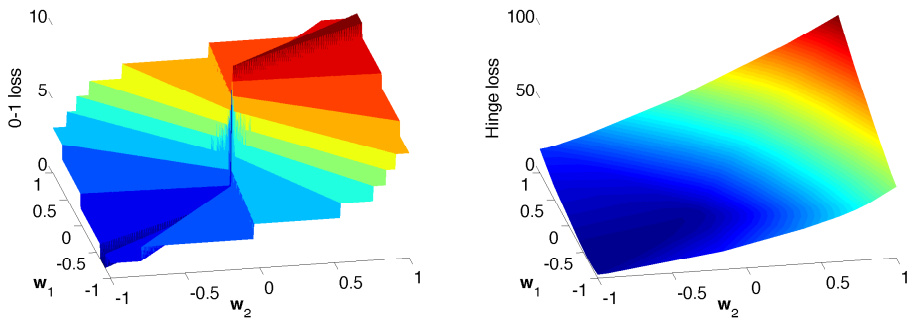



Figure 1.2 – *The empirical risk associated with the 0-1 loss (left) and the hinge loss (right) with respect to* $\boldsymbol{w}$ *on a linearly separable 2D toy dataset. The plots represent the value of both losses for every two-dimensional classifier* $\boldsymbol{w} = [\boldsymbol{w}_1, \boldsymbol{w}_2]^\top$ *with* $\boldsymbol{w}_1$ *and* $\boldsymbol{w}_2$ *in* $[-1; 1]$

Where, with a slight abuse of notation, we define $t_i \doteq t(\boldsymbol{x}_i)$.

Although this is a reasonable assumption in the general case, they may have some settings where the labels are not as readily available as the data. Actually, one of the motivations of this thesis is precisely to deal with those cases where the basic assumption that a training composed of reliable data-label pairs is available does not hold and we tackle this issue in the following chapters.

It occurs that sometimes it may be convenient to think of $\mathcal{S}$ differently. Namely, we may see each $\boldsymbol{x}_i$ and $t_i$ as a realization of some random variables $X$ and $T$ where $X \sim \mathscr{D}$. In this representation, $\mathcal{S}$ is akin to a collection of realizations of $X$ and $T$ drawn upon a joint distribution $\mathscr{D}_t$ such that $\mathbb{P}(T = t(X)|X) = 1$; and for the sake of concision, we will write:

$$\mathcal{S} \overset{\text{i.i.d}}{\sim} \mathscr{D}_t^N$$

**Risk**

With training sets comes the need to generalize the notion of *loss* to sets of examples. Such generalization is called *risk* as it aims to measure the average loss a given hypothesis will achieve on examples drawn from $\mathscr{D}$.

**Definition 1.4** (True Risk) *For a hypothesis $h \in \mathbb{H}$, an arbitrary concept $t$ and a fixed loss functional $l$ we define the* (true) risk *of $h$ as*

$$R^l_{\mathscr{D}_t}(h) \doteq \mathbb{E}_{\mathscr{D}_t}\left[l(h, X, T)\right]$$

Obviously, the *true risk* is an abstract measure of performance as we lack the knowledge of $\mathscr{D}_t$. Because of this, we have to rely on empirical estimations of the risk, based on finite observations of $\mathscr{D}_t$ such as $\mathcal{S}$.

**Definition 1.5** (Empirical Risk) *For a hypothesis $h \in \mathbb{H}$, a training set $\mathcal{S} \overset{i.i.d}{\sim} \mathscr{D}_t^N$ and a fixed loss functional $l$ we define the* empirical risk *of $h$ on $\mathcal{S}$ as*

$$R^l_{\mathcal{S}}(h) \doteq \frac{1}{N}\sum_{\mathcal{S}} l(h, \boldsymbol{x}_i, t_i)$$

Let us assume for a moment that we have an algorithm that, given $\mathcal{S}$, finds $h \in \mathbb{H}$ such that $h \doteq \arg\min_{\mathbb{H}} R^l_{\mathcal{S}}(h)$ for a fixed, convenient, loss $l$. The question of whether $R^l_{\mathscr{D}_t}(h)$ can be bounded in term of $R^l_{\mathcal{S}}(h)$ is of utmost importance as it will enable minimization of the *true risk* through minimization of the *empirical risk*. Nonetheless, designing such an algorithm —that is, one that provably finds a minimizer of $R^l_{\mathcal{S}}(.)$— is a problem on its own that we will tackle later in section 1.3.

### 1.2.3 P.A.C. learning and VC-dimension

Intuitively —and, to some extent, from the law of large number— it seems reasonable to think that if $\mathcal{S}$ is large enough then, for any $h \in \mathbb{H}$, the values of $R^l_{\mathcal{S}}(h)$ and $R^l_{\mathscr{D}_t}(h)$ will be close. In other words, the question of how *true* and *empirical* risks relate to each other seems a bit ill-posed. A far more interesting question, though, is to determine the minimal size of $\mathcal{S}$, if any exists, that guarantees an arbitrarily small distance between $R^l_{\mathcal{S}}(h)$ and $R^l_{\mathscr{D}_t}(h)$.

### P.A.C. learning

The *Probably Approximately Correct* setting (P.A.C.), or alternatively *Formal Setting*, is a framework aimed at providing formal and sound definitions to the fundamental notions in machine learning. Established by Valiant [Valiant, 1984] it is the cornerstone of modern learning theory and at its core, are the notions of *learning algorithm* and P.A.C.-*learnability*. In a nutshell, a *learning algorithm* $\mathcal{A}^l$ is an application from $(\mathbb{X} \times \mathbb{Y})^N$ to $\mathbb{H}$ such that, for a given training set $\mathcal{S}$ of size $N \doteq |\mathcal{S}|$:

$$\mathcal{A}^l(\mathcal{S}) \doteq \arg\min_{h \in \mathbb{H}} R^l_{\mathscr{D}_t}(h)$$

As for P.A.C.-*learnability* the following definition is given

**Definition 1.6** (P.A.C.-learnability) *Let $\mathbb{H} \subset \mathbb{Y}^{\mathbb{X}}$ some hypothesis class, $\mathbb{T} \subset \mathbb{Y}^{\mathbb{X}}$ some concept class and $\mathscr{D}$ an arbitrary distribution over $\mathbb{X}$. Moreover let $l$ some fixed loss function and $\delta, \epsilon \in ]0,1[$ some parameters set in advance. We say that $\mathbb{H}$ is* P.A.C.-*learnable assuming $\mathbb{T}$ if and only if: there exists a learning algorithm $\mathcal{A}^l$ and some integer $N_0(\epsilon, \delta)$ such that for any concept $t \in \mathbb{T}$ and training set $\mathcal{S} \overset{i.i.d}{\sim} \mathscr{D}_t^N$, $N \geq N_0(\epsilon, \delta)$ the following holds with probability at least $1 - \delta$:*

$$R^l_{\mathscr{D}_t}\left(\mathcal{A}^l(\mathcal{S})\right) \leq R^l_{\mathscr{D}_t}(h^*) + \epsilon$$

*Where $h^*$ is the* optimal hypothesis *of $\mathbb{H}$:*

$$h^* \doteq \arg\min_{h \in \mathbb{H}} R^l_{\mathscr{D}_t}(h)$$

This definition deserves some more exposition time to be fully grasped. First, remark that we have introduced two new parameters $\delta$ (confidence $1 - \delta$) and $\epsilon$ (accuracy $\epsilon$), hence the *Probably* THe reason for this is the necessity to allow for some degree of uncertainty in learning. The next observation is that we require to be close to the *optimal* solution $h^*$, it is important to note that, for now, no assumption has been made on $t$ and because of that, there is no guarantee that $R_{\mathscr{D}_t}(h^*) = 0$. Also, we have not yet restricted the execution time of $\mathcal{A}^l$ and the definition does not preclude exponential algorithms. If $\mathcal{A}^l$ is polynomial in $1/\delta$, $1/\epsilon$ and $N$ we will say that $\mathbb{H}$ is *efficiently* P.A.C.-learnable assuming $\mathbb{T}$. In practice though, $\mathcal{A}^l$ is generally polynomial in $N$ and $N_0(\epsilon, \delta)$ is a polynomial of $1/\epsilon$ and $1/\delta$.

The major caveat of P.A.C.-learnability lies in its requirement for algorithms that reliably produce almost optimal hypotheses, independently of $\mathcal{S}$ and $t$. With little other choice than directly minimizing $R^l_{\mathcal{S}}(h)$ we need a strong bond between empirical and true risks: we require that empirical and true risks are close for both $h^*$ and $h$, independently of $\mathcal{S}$ and $t$. Phrased differently, we do not only ask for $R^l_{\mathcal{S}}(h)$ to be close to $R^l_{\mathscr{D}_t}(h)$ with respect to some fixed $h$ but for any $h \in \mathbb{H}$ *at the same time*.

### VC-dimension

The VC-dimension was introduced by Vapnik and Chervonenkis in [Vapnik and Chervonenkis, 1971]. Not only they established theoretical

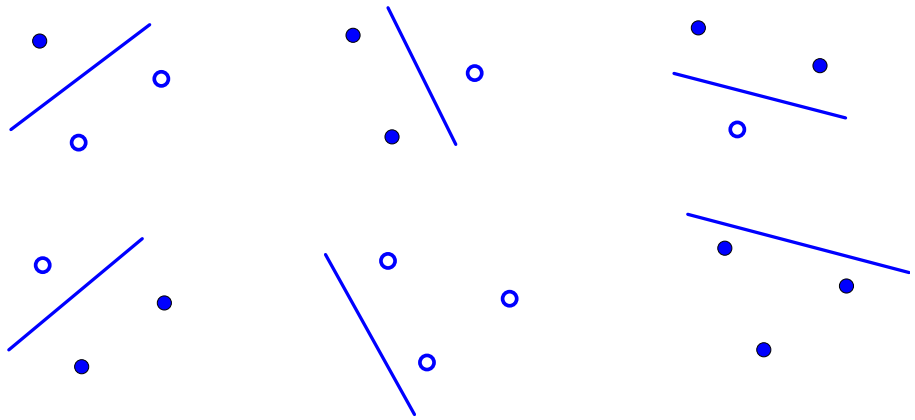


Figure 1.3 – *The case of hyperplane (with an offset term). Any of the* 6 *possible labellings over those* 3 *points can be produced by a linear classifier. Hence, the class shatters this set of* 3 *points.*

necessary conditions for $R^l_{\mathcal{S}}(h)$ to be close to $R^l_{\mathscr{D}_t}(h)$ but they also introduced a capacity measure that captures the notion of learnability. The theory of VC-dimension rests on the notion of capacity of a class of hypotheses $\mathbb{H}$. The starting idea is simple: the richer is $\mathbb{H}$, the larger is $N_0$ (see definition 1.6). Roughly, the intuition is that more diversity in $\mathbb{H}$ requires more observations —in other words elements in $\mathcal{S}$— to distinguish two hypotheses. While this is easily believable for finite hypothesis class where $|\mathbb{H}| < \infty$, VC-dimension happens to accuratly capture the 'effective' capacity of infinite hypothesis classes through the notion of *shattering*.

**Definition 1.7** (Shattering and VC-dimension) *For a hypothesis space* $\mathbb{H}$ *and a set of points* $\mathcal{E} \doteq \boldsymbol{x}_1, \dots \boldsymbol{x}_N$. *We say that* $\mathbb{H}$ shatters $\mathcal{E}$ *if and only if, for each subset* $\mathcal{E}' \in \mathcal{P}(\mathcal{E})$, *there exists an hypothesis* $h \in \mathbb{H}$ *such that*

$$h(\boldsymbol{x}) = +1 \Leftrightarrow \boldsymbol{x} \in \mathcal{E}'$$

*Additionally, the* VC*-dimension of* $\mathbb{H}$, *denoted* $\mathrm{VC}(\mathbb{H})$, *is defined as the size* $M \doteq |\mathcal{E}|$ *of the largest set* $\mathcal{E}$ *that is shattered by* $\mathbb{H}$. *If* $\mathbb{H}$ *can shatter arbitrarily large sets, then* $\mathrm{VC}(\mathbb{H}) \doteq \infty$.

The key observation behind this definition is that, for a given set $\mathcal{E}$ there is exactly $2^{|\mathcal{E}|}$ different labeling of $+1$ and $-1$. Intuitively, the VC-dimension corresponds to the critical moment when a hypothesis class is no longer rich enough to produce all of those $2^{|\mathcal{E}|}$ labellings.

**Theorem 1.1** (VC-dimension of linear classifiers) *Let* $\mathbb{H}$ *the hypothesis class of linear classifier of dimension* $D$, *as they were defined in definition 1.1 then* $\mathrm{VC}(\mathbb{H}) = D$[1].

Note that the class of linear classifiers is of particular interest here. Its VC-dimension scales linearly with the dimensionality of the data and

[1]Note that the usual result is for linear classifier with an offset term $b$. In this case, the VC-dimension is then equal to $D + 1$.

because of this linear classifiers represent the perfect trade-off between hypothesis classes that are too rich for learning to be relevant and the ones that are too poor for anything interesting to be learned. The proof of this theorem can be found in any Machine Learning textbook —e.g. chapter 13 of [Devroye et al., 1996]. A rough illustration of what the shattering property is also given in figure 1.3.

The VC-dimension plays a central role in machine learning essentially because of two fundamental results that underline the strong links between VC-dimension and P.A.C.-learnability. The first one directly link the VC-dimension to the notion of P.A.C.-learnability and lays the boundaries of what is learnable.

**Property 1.1** (Finite VC-dimension) *A hypothesis class* $\mathbb{H}$ *such that* $\mathrm{VC}(\mathbb{H}) = \infty$ *is not* P.A.C.-*learnable.*

The second results, stated here as a theorem, is what truly enables P.A.C.-learning as a valid theory of the learnable. In a nutshell, it links $N_0(\epsilon, \delta)$ —that is, the minimal size of $\mathcal{S}$, see definition 1.6— to the VC-dimension of a hypothesis class. Note that, for clarity consideration, and because doing otherwise would require to introduce a lot more definitions, we will only state a restricted version of this theorem. Namely, we will assume that $t \in \mathbb{H}$ or, alternatively, that $R^{l}_{\mathscr{D}_t}(h^*) = 0$. Additionally, the loss considered for this theorem is the 0-1-loss.

**Theorem 1.2** (Sample Complexity) *Let* $\mathbb{H}$ *a hypothesis class,* $\mathscr{D}$ *a distribution over* $\mathbb{X}$, $t \in \mathbb{H}$ *an arbitrary concept and* $\mathcal{S} \overset{i.i.d}{\sim} \mathscr{D}_t^N$ *a training set of size* $N$*. Let* $h \in \mathbb{H}$ *be a given hypothesis such that* $R^{l_{0-1}}_{\mathcal{S}}(h) = 0$*. Then, if* $N \geq N_0$ *with*

$$N_0 \in \mathcal{O}\left(\frac{1}{\epsilon}\log\left[\frac{1}{\delta}\right] + \frac{\mathrm{VC}(\mathbb{H})}{\delta}\log\left[\frac{1}{\epsilon}\right]\right)$$

*the following holds with probability* $1 - \delta$*:*

$$R^{l_{0-1}}_{\mathscr{D}_t}(h) \leq R^{l_{0-1}}_{\mathscr{D}_t}(h^*) + \epsilon$$

Since $R^{l_{0-1}}_{\mathscr{D}_t}(h^*) = 0$ it means that $R^{l_{0-1}}_{\mathscr{D}_t}(h) \leq \epsilon$. in plain English, the *Sample Complexity* theorem says that, if one has an algorithm that can find a hypothesis that does not err on $\mathcal{S}$, with $\mathcal{S}$ large enough, then $\mathbb{H}$ is P.A.C.-learnable (assuming $\mathbb{T} = \mathbb{H}$). The theorem can easily be generalized to the case where $\mathbb{T} \neq \mathbb{H}$, hence the importance of the 0-1-loss in its statement. Although it is an enthralling subject, VC-theory is not the focus of this thesis and it would impair our message to digress more lengthy on the subject, the interested reader can refer to [Shashua, 2009] for further readings though.

Last but not least, [Ben-David et al., 1995] and [Bartlett et al., 1994] have showed that similar results hold for other losses. Notably, those papers introduce a generalization of the VC-dimension, called *Fat-shattering dimension*, or *Pseudo-dimension* that applies to continuous loss functions. Note that a *sample complexity* result based on the pseudo-dimension is presented in Section 4.2, the proof given in Appendix A.2 follows the typical scheme of *sample complexity* proofs and, as such, is very close to the proof of, for instance, proposition 4.3.

## 1.3 A Few Examples of Machine Learning Methods

The previous sections of this chapter were mostly focused on exposing the fundamental results of learning theory are now ready to introduce some iconic methods of machine learning. The very purpose of a learning algorithm is to identify a hypothesis $h \in \mathbb{H}$ that fits some desirable criteria. An obvious one is, of course, to minimize the empirical risk, which is the cornerstone of the P.A.C.-setting. The first algorithm we will present — *the Perceptron Algortihm*— does exactly that, however we will also see that more refined criteria yield interesting results.

### 1.3.1 The Perceptron Algorithm

Perhaps one of the most well-known, and studied algorithms of all machine learning, the Perceptron algorithm is also one of the first linear classification methods —that is, methods focused on learning linear classifiers, see Definition 1.1. The idea of the Perceptron is a very simple, yet powerful, one: explore $\mathcal{S}$ one example at time, making locally optimal corrections when a mistake happens (see algorithm 1).

**Algorithm 1** An example of perceptron algorithm

**Require:** A sequence of points $(\boldsymbol{x}_1, t_1), \dots, (\boldsymbol{x}_N, t_N)$
1: $\boldsymbol{w}^0 \leftarrow 0$, $k \leftarrow 0$
2: **for all** $\boldsymbol{x}_i : i \in [N]$ **do**
3: **if** $t_i \left\langle \boldsymbol{w}^k; \boldsymbol{x}_i \right\rangle < 0$ **then** ▷ i.e. $h_{\boldsymbol{w}^k}(\boldsymbol{x}_i) \neq t_i$
4: $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + t_i \boldsymbol{x}_i$
5: $k \leftarrow k + 1$
6: **end if**
7: **end for**

The perceptron algorithm belongs to the family of online algorithms, those are algorithms that do not require a full access to $\mathcal{S}$ at all time but instead process the elements of $\mathcal{S}$ one after another independently. Online algorithms have the invaluable advantage that they can run sequentially: stopped and restarted at will. For instance, $\mathcal{S}$ may be generated *on the fly* as the algorithm is able to wait until new examples are available.

That being said, the most interesting property of the Perceptron resides in its convergence behaviour. While the algorithm was first introduced in [Rosenblatt, 1958] its convergence has remained unproved for a few years until *Block* and *Novikoff* proposed the following theorem.

**Theorem 1.3** (Convergence of Perceptrons [Block, 1962, Novikoff, 1962]) *Let $(\boldsymbol{x}_1, t_1), \dots, (\boldsymbol{x}_N, t_N)$ be any sequence of points in $\mathbb{X} \times \mathbb{Y}$ such that $\|\boldsymbol{x}_i\| \leq R$ for all $i$. If there exists a classifier $\boldsymbol{w}^*$ of norm $\|\boldsymbol{w}^*\|_2 = 1$ such that $\gamma^{\boldsymbol{w}^*} \doteq \min_{i \in [N]} \gamma_{\boldsymbol{x}_i}^{\boldsymbol{w}^*}$ then, the Perceptron algorithm makes at most*

$$\frac{R^2}{\left(\gamma^{\boldsymbol{w}^*}\right)^2}$$

*updates and outputs a classifier $\boldsymbol{w}$ that makes no mistake on the sequence $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$.*

A few things should be said on the subject of this theorem. First, note that we introduce the notion of margin *over an entire set* ($\gamma^{w^*}$); notion that is formalized for any training set in definition 1.8.

**Definition 1.8** (Margin of a Set) *Let set $\mathcal{S}$ a training set and $h_w \in \mathbb{H}$ a linear classifier. We define the margin $\gamma^w_{\mathcal{S}}$ of $h$ on $\mathcal{S}$ as:*

$$\gamma^w_{\mathcal{S}} \doteq \min_{x_i \in \mathcal{S}} \gamma^w_{x_i}$$

But, more importantly, it is the scope of this theorem that is remarkable, as this result holds independently of the underlying distribution. In other words, the points $x_1, \ldots, x_N$ of the sequence are not required to follow any form of distribution. Notably, when combined with the fact that the perceptron is an online algorithm, one has a powerful convergence result that holds even when the sequence $x_1, \ldots, x_N$ is computed after each step $k$ not only dynamically, but with respect to the current Perceptron's solution $w_{k-1}$ —see, algorithm 1. This is perhaps the most underexploited property of the perceptron algorithm; this very idea will resurface at different points in this thesis and will prove an useful trick throughout the next chapters. Finally, even though we will not provide the proof for this theorem, the reader may still refers to the proof Th 2.3 in appendix A.1.1 which is an extension of this result and, when applied to the perceptron, reduces to theorem 1.3.

### 1.3.2 (Hard Margin) Support Vector Machines

More than a Machine Learning algorithm, Support Vector Machines — SVM for short— are a way to recast the problem of learning linear classifiers into a more different framework than the one of empirical risk minimization. Introduced by Vapnik in [Boser et al., 1992], SVMs take their roots in the VC-theory of linear classifier.

In itself theorem 1.1 is not undesirable and as discussed above the VC-dimension of linear classifiers grows linearly with respect to the dimension of $\mathbb{X}$. There are nonetheless two reasons that motivate the need for a better result. First, linear classifiers are very limited in term of what concept they can model in low dimension, and practical uses of linear methods involve very high dimensional datasets. Second, the solution $h^*$ defined by the empirical risk minimization strategy is, more than often, not unique, thus the problem seems ill-posed. Remedying to these issues is achieved through regularization. Namely, we will reformulate the original risk minimization problem in such a way that it has a better defined solution. Practically speaking this is done by asking for the final solution to fulfill some additional properties based on its margin. Theoretically speaking, doing so does not go against P.A.C.'s empirical risk minimization principle though and can be thought of as artificially shrinking the hypothesis space. Hence, from this perspective, SVMs simply work on a hypothesis class that is only a subset of all linear classifiers.

The SVM paradigm casts the problem of learning linear classifier in terms of finding a *large-margin classifier*. In other words, for a given dataset $\mathcal{S}$, SVMs look for a constrained version of $h^*$:

$$h^*_{\text{SVM}} \doteq \arg\max_{h \in \mathbb{H}} \gamma^h_{\mathcal{S}}$$

Another way to think of this is simply as a qualitative ordering over all the possible $h^*$, where the picked solution is obviously the best one; hence, it should be noted that $h^*_{\text{SVM}}$ is, per se, a valid candidate for $h^*$.

More precisely, the canonical, equivalent, formulation of the SVM problem is as follow:

$$\text{Find } \boldsymbol{w} = \arg\min \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{s.t.} \quad \forall \boldsymbol{x}_i \in \mathcal{S} : t_i \langle \boldsymbol{w}; \boldsymbol{x}_i \rangle \geq 1 \tag{1.1}$$

This latter form is actually a *quadratic programming* problem, thus any quadratic optimization algorithm can directly solve a SVM problem in this form.

Not only SVMs introduce an slightly different problem, but, as hinted before, they also allow some control over the VC-dimension of the solution as it is shown by the next theorem:

**Theorem 1.4** (VC-dimension of large margin classifier [Vapnik, 1998]) *Let $\mathcal{E}$ a* working set *of possible input such that $\max_{\mathcal{E}} \|\boldsymbol{x}\| \leq R$, and $\mathbb{H}^{\mathcal{E}}_T$ a hypothesis class of linear classifier such that for all $\boldsymbol{w} \in \mathbb{H}^{\mathcal{E}}_T$:*

- $\|\boldsymbol{w}\|_2 \leq T$
- $\gamma^{\boldsymbol{w}}_{\mathcal{E}} \geq 1$

*Then, the* VC*-dimension of $\mathbb{H}^{\mathcal{E}}_T$ is*

$$\text{VC}\left(\mathbb{H}^{\mathcal{E}}_T\right) = R^2T^2$$

The notion of maximum margin is tied, as seen before, with the norm of $\boldsymbol{w}$ under the constraint of a —at least— unitary margin. For the punctilious reader, this theorem may be quite unsatisfactory because the VC-dimension is now tied to some working set of points, $\mathcal{E}$, that we could not know in advance. Indeed $\mathcal{E}$ must contains not only $\mathcal{S}$ but all the points that are likely to be drawn from $\mathscr{D}$ for the subsequent predictions; in other words, knowing $\mathcal{E}$ in advance amounts to knowing $\mathscr{D}$. Nevertheless, this discussion, albeit interesting, is not central to this thesis and the interested reader shall refer to [Mount, 2015] for a more in depth debate on the VC-dimension of large margin classifiers.

More generally, SVM represent a paradigm shift in machine learning. Instead of simply minimizing the empirical risk, the focus is now onto the simultaneous minimization of the empirical risk *and* the VC-dimension. A strategy usually referenced as *structural risk minimization* in opposition to the previous *empirical risk minimization*. We will see later —i.e. Section 2.1— how this new paradigm was pivotal in the evolution of machine learning and why, consequently, structural risk minimization has established itself as the privileged approach for linear classifier learning.

## 1.4 Conclusion

Throughout this chapter, we have provided a snapshot of what is perhaps the most studied problem of machine learning: binary classification. It goes without saying that this exposition is by no means supposed to

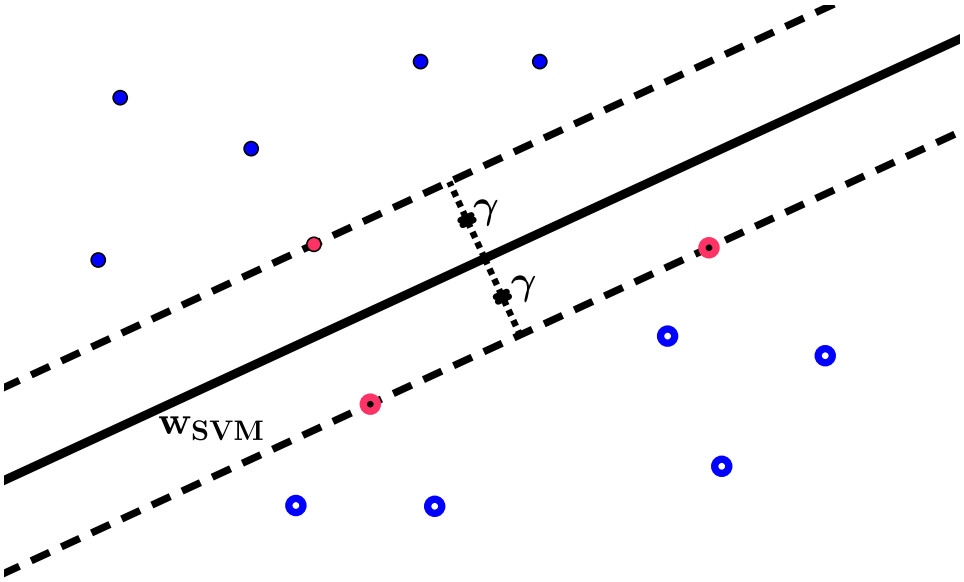



Figure 1.4 – *An example of SVM classifier. The filled points (resp. empty) correspond to positive (resp. negative) examples. The classifier achieve a margin $\gamma$ on the dataset, and the three red points are the support vectors, that is the datapoint of minimal margin.*

be exhaustive and a lot of both fundamental and brilliant works were skipped. Nonetheless, we have reviewed the fundamental notions upon which are built this thesis and all of the algorithms and notions exposed later ultimately stem from the notions and algorithms introduced here.

# Some Extensions to Classification

2

Contents



The setting we have described so far —i.e. linear binary classification— may be appealing by its simplicity but it is also limited in its expressivity. In this chapter, we shall provide the tools and means for the last chapter's notions to be relevant in a practical context. Namely, we will introduce the notion of kernel that will allow for non-linear classifiers to be trained within the framework of linear classification. Some of the implications of using kernels will be discussed in section 2.2 where we shall briefly mention the matter of sparsity through Sample Compression Scheme. Additionally, the last section is devoted to expanding our setting beyond discriminating between positive and negative data by allowing problems with an arbitrary number of classes, in other words, multiclass learning problems.

In many aspects, this chapter plays the role of a wrapper regarding the last one: it allows for simple and theoretically sound concepts to be fully taken advantage of in a complex environment.

## 2.1 Kernels, or the true power of linear classifier

### 2.1.1 From Input space to Feature Space

So far, we have limited ourselves to linear classifiers. While the theory about those is arguably well established, one must reckon that they lack the expressive power of more sophisticated hypothesis classes. Indeed, a linear classifier simply computes a weighted sum associated with a fixed threshold to decide the class of a point. Building on this idea, the most immediate step in the direction of more expressive classifiers is to augment datapoints by adding new features derived from non-linear combinations of the original ones. Therefore linear classifiers would gain in expressivity through the use of these new, augmented, features.

Kernels in machine learning do just that. They allow to work with augmented learning examples by adding new, features derived from the existing ones. The obvious benefit of this is that linear classifiers are still a relevant hypothesis class in this setting, albeit a lot more expressive. More formally, we define the *feature space* $\mathbb{F}$ with inner product $\langle \cdot ; \cdot \rangle_{\mathbb{F}}$ and the *feature map*, a function $\phi : \mathbb{X} \to \mathbb{F}$ which maps each point of $\mathbb{X}$ to a point in $\mathbb{F}$. Therefore, a (linear) classifier in the feature space is akin to a hyper-plane in $\mathbb{F}$.

The crux of kernel methods is the existence of a *Mercer kernel functions*.

**Definition 2.1** (Mercer Kernel) *A function $k : \mathbb{X} \times \mathbb{X} \mapsto \mathbb{R}$ is a Mercer Kernel if and only if it is both symmetric and semi-definite positive:*

- $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{X} : k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\boldsymbol{x}_2, \boldsymbol{x}_1)$
- *For all $N \in \mathbb{R}_+$, sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ of elements in $\mathbb{X}$ and all $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N$ in $\mathbb{R}$:*

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

The existence of such kernels allows the use of the *kernel trick* thanks to the *Mercer's theorem*.

**Theorem 2.1** (Mercer's theorem [Mercer, 1909]) *Let $\mathbb{X}$ an input space and $k$ a Mercer's kernel, that is $k$ is symmetric and semi-definite positive. Then, there exists a feature map $\phi : \mathbb{X} \mapsto \mathbb{F}_\phi$, with $\mathbb{F}_\phi$ a Hilbert's space, such that, for all $\boldsymbol{x}, \boldsymbol{x}'$ in $\mathbb{X}$:*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \phi(\boldsymbol{x}); \phi(\boldsymbol{x}') \right\rangle_{\mathbb{F}_\phi}$$

This theorem means that, as long as a Mercer kernel exists, one can work with transformed data without directly dealing with their representation in the feature space, that may be of infinite dimension. This particular property is referred in the literature as the *kernel trick* and allows for linear classifiers to model any arbitrary concept. A notable special case is when $\mathbb{F}$ is of infinite dimension, in that case, according to theorem 1.1, the VCdimension of linear classifiers would grow towards infinity, which would imply that proper learning is no longer be possible. Hence the necessity of paradigms such as structural risk minimization and the widespread use of SVM methods in machine learning which integrate in

their design a way to control the VC-dimension of $\mathbb{H}$ by making it independent of the dimensionality of $\mathbb{F}$ (see Th. 1.4). Consequently, methods such as SVM can work with infinite feature space while keeping the VC-dimension of $\mathbb{H}$ in check, allowing both generalization guarantees and expressive hypothesis classes.

**Example 2.1** (Polynomial Kernel) Polynomial kernels *are a class of Mercer's kernel defined as:*

$$k^{a,b}_{poly}(\boldsymbol{x}, \boldsymbol{x}') \doteq (\langle \boldsymbol{x}; \boldsymbol{x}' \rangle + a)^b = \left( a + \sum_{i=1}^{D} \boldsymbol{x}_i \boldsymbol{x}'_i \right)^b$$

*with $a \geq 0$ and $b > 0$.*

*Once expanded, the number of terms of this polynomial is given by the* multinomial theorem *and it is easy to see that a feature space $\mathbb{F}$ such that $\langle \phi(\cdot); \phi(\cdot) \rangle_{\mathbb{F}} = k^{a,b}_{poly}(\cdot, \cdot)$ would require one dimension per expanded term.*

**Example 2.2** (Gaussian Kernel) Gaussian Kernels *(or, sometimes,* Radial Basis Function (RBF) kernels*) are another popular class of Mercer's kernel defined as:*

$$k^{\sigma}_{RBF}(\boldsymbol{x}, \boldsymbol{x}') \doteq \exp\left( -\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\sigma^2} \right)$$

*It is known that the corresponding feature space $\mathbb{F}$ is then of infinite dimension —see, e.g. [Shashua, 2009]. An interesting property of these kernels is that $k^{\sigma}_{RBF}(\boldsymbol{x}, \boldsymbol{x}) = 1$; therefore for any point $x \in \mathbb{X}$, $\|\phi(\boldsymbol{x})\|_2 = 1$.*

### 2.1.2 Learning in Feature Space

Kernels seem so far to preclude any direct use of the projected data. Indeed, the representations $\phi(\boldsymbol{x}_i)$ in the feature space are possibly of infinite size and could not be handled by learning algorithms, the same also applies to classifiers. More importantly, because of the kernel trick, those representations and the feature space may not even be explicitly known. One way to circumvent this problem by rewriting algorithms without explicit mention of $\boldsymbol{w}$ or $\phi(\boldsymbol{x}_i)$. While the formal exposition of this idea might overburden this thesis with notions and results otherwise ancillary to the main topic, we might still give the general intuition through a case example for the Perceptron algorithm.

**Example 2.3** (Kernel Perceptron) *Let define $\boldsymbol{w}$ the classifier that would be learned by the perceptron algorithm on a feature space $\mathbb{F}$. From the Perceptron algorithm (see Alg. 1) we have that:*

$$\boldsymbol{w} = \sum_{i=1}^{M} \phi(\boldsymbol{x}^i_{up})$$

*where $\boldsymbol{x}^1_{up}, \ldots, \boldsymbol{x}^M_{up}$ is the sequence of datapoints erred upon during the execution of the algorithm. Note that, for simplicity, we allow duplicate in this sequence as this correspond to the case where the Perceptron makes repetitive mistakes on the same point. From this, the decision rules associated with $\boldsymbol{w}$ for a new point $\boldsymbol{x}'$ can be rewritten as the sign of:*

$$\langle \boldsymbol{w}; \phi(\boldsymbol{x}') \rangle_{\mathbb{F}} = \sum_{i=1}^{M} \left\langle \phi(\boldsymbol{x}^i_{up}); \phi(\boldsymbol{x}') \right\rangle_{\mathbb{F}} = \sum_{i=1}^{M} t(\boldsymbol{x}^i_{up}) k(\boldsymbol{x}^i_{up}, \boldsymbol{x}')$$

*Hence, we do not need an explicit representation of $\boldsymbol{w}$ or $\phi(\boldsymbol{x}')$ to make predictions. The same idea extends to the whole Perceptron algorithm and one can rewrite it uniquely in terms of the Mercer's kernel $k$.*

As advertised before, the general case (notably for SVM) is more tricky as one has to ensure that there exists such decomposition of $\boldsymbol{w}$ in the feature space. The interested reader may refer to the *representer theorem* (see, e.g. [WAHBA, 1990, Schölkopf et al., 2001, Vert et al., 2004]) which establishes that such representation exists.

More precisely, the SVM problem can be rewritten in its so-called *dual* form:

$$\max_{\boldsymbol{\alpha}_i, i\in[N]} \sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i,j}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j t_i t_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \text{s.t.} \quad \forall i \in [N]\ \boldsymbol{\alpha}_i \geq 0 \tag{2.1}$$

and the decision rules now become:

$$h^*_{\text{SVM}}(\cdot) = \text{sign}\left(\sum_{i=1}^{N} \boldsymbol{\alpha}_i k(\boldsymbol{x}_i, \cdot)\right)$$

with $N$ the size of the training set $\mathcal{S}$ which contains the $\boldsymbol{x}_i$ and $t_i$ and the number of parameters $\boldsymbol{\alpha}_i$ thus scale linearly with the number of data in $\mathcal{S}$.

## 2.2 Compression Scheme

### 2.2.1 A motivation for Sample Compression Scheme

With the use of kernels (see previous section), $\boldsymbol{w}$ can no longer be manipulated as a vector, and we have instead to rely on a representation based on the inner product in $\mathbb{F} : \langle \cdot ; \cdot \rangle_{\mathbb{F}}$. More precisely, in most cases, given a training set $\mathcal{S}$ of size $N$:

$$\langle \boldsymbol{w}; \cdot \rangle_{\mathbb{F}} = \sum_{i=1}^{N} \boldsymbol{\alpha}_i k(\boldsymbol{x}_i, \cdot)$$

where the values of the $\boldsymbol{\alpha}_i$ depend on the learning algorithm.

For instance, in the SVM case, the $\boldsymbol{\alpha}_i$ are set by solving optimization problem (2.1) but, on the other hand, for the Perceptron case, they are defined by the mistakes made during the learning process. The end result is the same: computing the decision rule is now tied to the size $N$ of the training set, making large training sets a burden for later predictions. One feature of SVM that was instrumental in their success is that SVM actually tend to produce *sparse* solutions (see also figure 2.1). That is, only a few of the $\boldsymbol{\alpha}_i$ are in fact nonzero and, as such, the vector $\boldsymbol{\alpha}$ is therefore *sparse*. For any dataset $\mathcal{S}$, there is only a limited number of points that lie at the edge of the margin of the SVM classifier —i.e. there are only a few points $\boldsymbol{x}$ verifying $\gamma_{\boldsymbol{x}}^{h^*_{\text{SVM}}} = \gamma_{\mathcal{S}}^{h^*_{\text{SVM}}}$. Moreover, those points alone are sufficient to define $h^*_{\text{SVM}}$ —one simply has to solve a SVM problem on those sole points— and, when working with a full dataset, solving the optimization problem underlying the SVM formulation amounts to identifying those points. To some extends, Perceptrons have a similar feature, in the sense that only

the points that caused some errors have non-zero $\boldsymbol{\alpha}$ value. However those points are far more numerous than in the SVM's case.

All of this points toward the idea that a training set $\mathcal{S}$ is, from a learning perspective, redundant in most cases. The driving idea behind *Sample Compressions Scheme* is precisely to analyze the benefit of being able to capture all the information in a small number of datapoints.

### 2.2.2 Sample Compression Scheme and Results

**Definition 2.2** (Sample Compression Scheme [Littlestone and Warmuth, 1986]) *Let denote by* $\mathbb{S}(m) = (\mathbb{X} \times \mathbb{Y})^m$ *the set of all training sets of size m, then, a Sample Compression Scheme of size L consists in a pair of mappings*

$$\kappa : \bigcup_{m=L}^{\infty} \mathbb{S}(m) \to \mathbb{S}(L)$$

*and*

$$\rho : \bigcup_{m=L}^{\infty} \mathbb{S}(m) \to \mathbb{Y}^{\mathbb{X}}$$

*such that for any* $\mathcal{S} \sim \mathscr{D}_t^N$ *with arbitrary t and N and any* $\boldsymbol{x} \in \mathbb{X}$*:*

$$\kappa(\mathcal{S}) \subseteq \mathcal{S} \quad \textit{and} \quad \rho(\mathcal{S}) = \rho(\kappa(\mathcal{S}))$$

As previously hinted, both SVM methods and Perceptrons fall in the definition of *Sample Compression Scheme*. For a given classifier $\boldsymbol{w}$ taken in kernel form —that is, $\boldsymbol{w} = \sum_{i=1}^{N} \boldsymbol{\alpha}_i \phi(\boldsymbol{x}_i)$ where the vector $\boldsymbol{\alpha}$ is determined by the learning algorithm— $\kappa(\mathcal{S})$ is the list of examples $\boldsymbol{x}_i$ for which $\boldsymbol{\alpha}_i \neq 0$ while $\rho$ is the learning procedure itself that learns $h_w$ from those examples.

Similarly to the VC-dimension for hypothesis classes, the *size* $|kappa(\mathcal{S})|$ of a compression scheme allows for some control over the variety of classifier that $\rho$ can output. The idea being that only a handful of classifiers can be learned from a small *compressed* dataset $\kappa(\mathcal{S})$. In fact, the connection is more profound and it has been shown that if a hypothesis class can be learned from a Sample Compression Scheme then it is P.A.C.-learnable and thus it has a finite VC-dimension [Littlestone and Warmuth, 1986]. Moreover, [Moran and Yehudayoff, 2015] have recently proved the converse to be true: finite VC-dimension implies the existence of a Sample Compression Scheme.

From here, it is only natural that Sample Compression Schemes provide generalization guarantees akin to the VC ones. Namely, we refer to the case where one has a training set $\mathcal{S}$ of size $N$ and learn from a compressed set $\kappa(\mathcal{S})$ of size $L$.

**Theorem 2.2** (Sample Complexity for Sample Compression Schemes [Floyd and Warmuth, 1995]) *Let* $\mathbb{H}$ *a hypothesis class with Sample Compression Scheme of size at most L,* $\mathscr{D}$ *a distribution over* $\mathbb{X}$*,* $t \in \mathbb{H}$ *an arbitrary concept and* $\mathcal{S} \overset{i.i.d}{\sim} \mathscr{D}_t^N$ *a training set of size N. Let* $h \doteq \rho\kappa(\mathcal{S})$*. Then, if* $N \geq N_0$ *with*

$$N_0 \in \mathcal{O}\left(\frac{1}{\epsilon}\log\left[\frac{1}{\delta}\right] + L + \frac{L}{\epsilon}\log\left[\frac{1}{\epsilon}\right]\right)$$

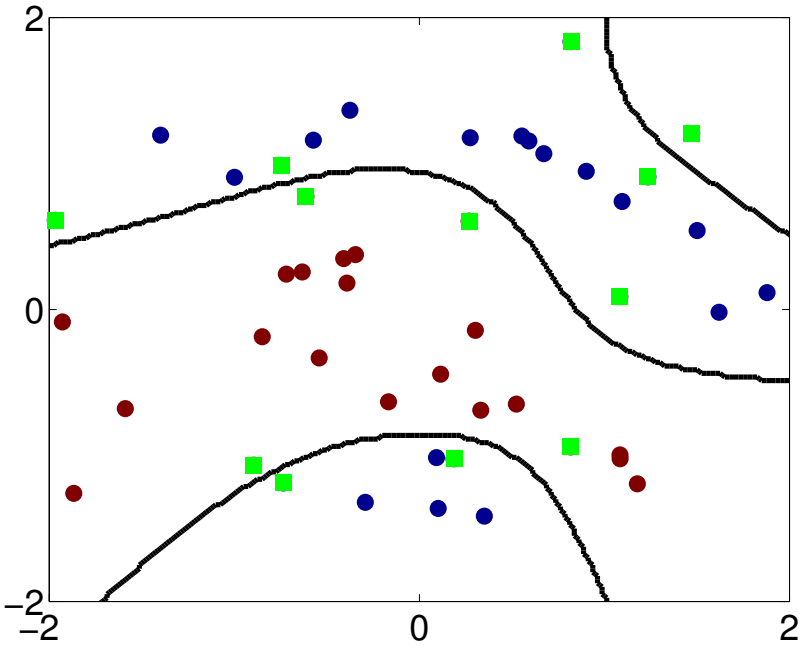



Figure 2.1 – *A kernelized SVM classifier. The red and blue dots are positive and negative examples. The decisions boundary of the classifier is drawn in black and the green squares correspond to the Support Vectors. In this example, those* 15 *support vectors are enough to characterize the SVM's solution whereas the full dataset is composed of* 50 *data.*

*the following holds with probability* $1 - \delta$*:*

$$R^{l_{0-1}}_{\mathscr{D}_t}(h) \leq R^{l_{0-1}}_{\mathscr{D}_t}(h^*) + \epsilon$$

This is but a snapshot of the results available for Sample Compression Schemes, and far more specific bounds exist for various settings. We refer the interested readers to the relevant literature, in particular, the results of [Littlestone and Warmuth, 1986, Floyd and Warmuth, 1995, Graepel et al., 2005].

## 2.3 Multiclass Classification

Until now, we have limited ourselves to the setting of binary classification, that is $\mathbb{Y} \doteq \{-1, 1\}$. This section is devoted to the so-called *Multiclass Classification* problem. More precisely, we focus in this section output spaces such as

$$\mathbb{Y} \doteq \{1, \ldots, Q\}$$

where $Q$, the number of classes, is specific to each problem and is known in advance.

This is a widely studied and important topic of Machine Learning, both for practical and theoretical reasons, and this section should not be thought as an overview but merely as an introduction to the ideas that will be relevant to the remaining of this work.

### 2.3.1 The Basics: OVA and OVO

The most natural idea to perform Multiclass Classification is to map the problem to a collection of binary learning tasks and then to solve them with the usual, bi-class, methods. Namely, we we discuss two particular ways to achieve such mapping.

- The *One-vs-All (*OVA*)* approach, which consists in learning $Q$ bi-class classifiers (one for each class) that are each able to predict the corresponding class membership —i.e. the classifier $i$ predicts whether or not a datapoint is in class $i$.
- The *One-vs-One (*OVO*)* approach, which relies on learning $Q(Q-1)/2$ classifiers (one per class dichotomy) that predicts, for two given classes $i$ and $j$ whether a point is more likely to belong to $i$ or $j$. The predicted class is then the one that 'wins' the most dichotomies.

Both of those ideas have their perks and constraints. Among the most obvious ones we can mention that OVA actually has to be used with classifiers that not only output a binary decision but also a confidence rating —e.g. the margin $\gamma$ for linear classifier— because multiple OVA classifiers can predict membership to different classes. Despite that not being a problem with OVO, one still has to come with a solution for cases where two classes are tied. Computationally speaking, the two methods are also very different, and depending on one's data, binary algorithm and general setting, either OVA or OVO may be the more efficient approach. In the end, the bottom line is this: in spite of their simplicity, OVA and OVO remains two very efficient approach to tackle multiclass problems and, in practice, those two very naive ideas yield state-of-the-art results [Li et al., 2005].

### 2.3.2 Ultraconservative Algorithms

Although the literature related to Multiclass Classification is very dense, the part of this thesis that relates to this setting takes its roots into one fundamental work proposed by Crammer and Singer [Crammer and Singer, 2003]. This paper notably avoid the mapping solution and instead tackle Multiclass Classification upfront, as one unitary problem. The result is an elegant generalization of the Perceptron algorithm that simultaneously learn $Q$ interdependent classifiers.

More formally, they give the following definition of *multiclass linear classifier*, that we will also use throughout this thesis:

**Definition 2.3** (Multiclass Linear Classifier) *We call* multiclass linear classifier *of matrix* $\mathbf{W} \in \mathbb{R}^{D\times Q}$*, the function* $h_{\mathbf{W}} \in \mathbb{Y}^{\mathbb{X}}$ *such that for any point* $\boldsymbol{x} \in \mathbb{X}$*:*

$$h_{\mathbf{W}}(\boldsymbol{x}) \doteq \arg\max_{i\in\mathbb{Y}} \langle \mathbf{W}_{.i}; \boldsymbol{x}\rangle \tag{2.2}$$

*where* $\mathbf{W}_{.i}$ *refers to the ith row vector of the matrix* $\mathbf{W}$*.*

Additionally, if there exist two solution $i$ and $j$ to problem (2.2) such that $\langle \mathbf{W}_{.i}; \boldsymbol{x}\rangle = \langle \mathbf{W}_{.j}; \boldsymbol{x}\rangle$ then we just one arbitrarily (say, e.g., the lower one).

Hence, multiclass linear classifiers are defined as matrices in $\mathbb{R}^{D\times Q}$. However, note that unlike their bi-class counterpart, the row vectors of $\mathbf{W}$ are more akin to (rescaled) barycenter than separating hyperplane in the sense that for each class, the corresponding row vector of $\mathbf{W}$ will be pointing towards the barycenter of the subset of example belonging to that class. Moreover, each of those vectors are in relation with the others and any rescaling operation must be applied to $\mathbf{W}$ as a whole in order to keep the different row vector balanced with each other. Notably, this is not the case in binary classification —and, by extension to OVA and OVO methods— where one can rescale each $\boldsymbol{w}$ seamlessly.

Thus, the problem of Multiclass Classification amounts to learn a matrix $\mathbf{W}$ that minimizes the empirical risk of $\mathcal{S}$, a problem similar to the one tackled by the Perceptron algorithm. In their paper ([Crammer and Singer, 2003]) Crammer and Singer not only propose a generalization of the Perceptron algorithm to multiclass problems, but a whole new class of algorithms that encompass both bi-class and multiclass Perceptrons while retaining the convergence properties previously stated by Block and Novikoff [Block, 1962, Novikoff, 1962]. They refer to this family of algorithm as *Ultraconservative Additive Algorithms* and provide the algorithmic scheme depicted in Alg. 2. The term *ultraconservative* refers to the fact that only those prototype vectors $\mathbf{W}_{\cdot r}$ which achieve a larger inner product $\langle \mathbf{W}_{\cdot r}; \boldsymbol{x}\rangle$ than $\left\langle \mathbf{W}_{\cdot t(\boldsymbol{x})}; \boldsymbol{x}\right\rangle$, that is, the vectors that can entail a prediction mistake when decision rule in equation (2.2) is applied, may be affected by the update procedure. The term *additive* conveys the fact that the updates consist in modifying the weight vectors $\mathbf{W}_{\cdot r}$'s by adding a portion of $\boldsymbol{x}$ to them (which is to be opposed to multiplicative update schemes).

**Algorithm 2** The family of Ultraconservative Additive Algorithm

**Require:** A sequence of points $(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)$

$\mathbf{W}^0 \leftarrow 0$, $k \leftarrow 0$

**for all** $\boldsymbol{x}_i : i \in [N]$ **do**

  $\mathcal{E} \doteq \{r \neq t_i : \langle \mathbf{W}^k_{\cdot r}; \boldsymbol{x}_i\rangle \geq \left\langle \mathbf{W}^k_{\cdot t_i}; \boldsymbol{x}_i\right\rangle\}$

  **if** $\mathcal{E} \neq \emptyset$ **then**

    Pick $\tau^k_1, \ldots \tau^k_Q$ such that:

$$\sum_{r=1}^{Q} \tau^k_r = 0 \tag{2.3}$$

$$\tau^k_{t_i} = 1 \tag{2.4}$$

$$r \notin \mathcal{E} \cup \{t_i\} \Rightarrow \tau^k_r = 0 \tag{2.5}$$

    $\forall r \in [Q] : \mathbf{W}^{k+1}_{\cdot r} \leftarrow \mathbf{W}^k_{\cdot r} + \tau^k_r \boldsymbol{x}_i$

    $k \leftarrow k + 1$

  **end if**

  **return** $\mathbf{W}^k$

**end for**

The general scheme of the algorithm follows the one of the Percep-

tron, with $\mathbf{W}$ being updated only when a mistake happens. The novelty here being the set $\mathcal{E}$ and the parameter $\tau$. The set $\mathcal{E}$ is basically an error set; it contains the classes that would be ranked higher than the true class in equation (2.2) of Definition 2.3. The parameter $\tau$ controls the update behaviour of the algorithm and different $\tau$ lead to different Ultraconservative Additive Algorithms. As such, there is a few things worth noting about $\tau$. In a nutshell, $\tau$ dictates how the classes in $\mathcal{E}$ will affect the update step; in this regard, the only rule is that, in the end, the total weights over the classes of $\mathcal{E}$ is equal to $-1$. In particular, not every classes in $\mathcal{E}$ are required to have a corresponding negative value in $\tau$. In fact, it is quite natural to just pick one class $r$ and set the corresponding $\tau_r$ to $-1$ and all other value to 0 —except of course for $\tau_{t_i}$—, the class $q$ may not even be the class (wrongly) predicted by $\mathbf{W}$ but any arbitrary class in $\mathcal{E}$. Finally, note that the value of $\tau$ does not need to be the same from one iteration to another, which ultimately allows for adaptive learning schemes.

The strength of Ultraconservative Additive Algorithms is that despite all the freedom left on $\tau$ they still enjoy a mistakes bound similar to the Perceptron's one, independently of $\tau$'s policy.

**Theorem 2.3** (Mistakes Bound For Ultraconservative Additive Algorithm) *Let $(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)$ be any sequence of points in $\mathbb{X} \times \mathbb{Y}$ such that $\|\boldsymbol{x}_i\| \leq R$ for all $i$. If there exists a multiclass classifier $\mathbf{W}^*$ of norm $\|\mathbf{W}^*\|_F$ such that*

$$\gamma^{\mathbf{W}^*} \doteq \min_{i \in [N]} \{ \langle \mathbf{W}^*_{\cdot t_i}; \boldsymbol{x}_i \rangle - \max_{r \neq t_i} \langle \mathbf{W}^*_{\cdot r}; \boldsymbol{x}_i \rangle \}$$

*then, any Ultraconservative Additive Algorithm make at most*

$$\frac{2R^2}{\left(\gamma^{\mathbf{W}^*}\right)^2}$$

*updates and output a classifier $\mathbf{W}$ that make no mistake on the sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$.*

Where the Frobenius norm $\| \cdot \|_{\mathrm{F}}$ is defined as:

**Definition 2.4** (Frobenius Norm) *Let $\mathbf{M}$ a $D \times Q$ matrix. Then the* Frobenius *norm of $\mathbf{M}$, written $\|\mathbf{M}\|_F$ is defined as:*

$$\|\mathbf{M}\|_F \doteq \sqrt{\sum_{i=1}^{Q} \sum_{j=1}^{D} \left(\mathbf{M}_{ij}\right)^2}$$

Also, note that the proof of theorem 2.3 is given as an example of mistakes bound's proof in appendix A.1.1.

### 2.3.3 A More General Formalization of Multiclass Classification

Before closing this section we shall mention another, strictly more general, formalization of the Multiclass setting.

We introduce a *feature vector representation* function $\Psi : \mathbb{X} \times \mathbb{Y} \mapsto \mathbb{X}^Q$ such that, for any point $\boldsymbol{x} \in \mathbb{X}$ of class $t(\boldsymbol{x})$, $\Psi(\boldsymbol{x}, t(\boldsymbol{x}))$ is a vector composed of $Q$ blocks of size $D$ with the $t(\boldsymbol{x})$'th block equal to $\boldsymbol{x}$ and the others set to 0 everywhere.

The multiclass algorithms discussed above can be used verbatim with this representation, given that one uses the vectorization of $\mathbf{W}$: $\text{vect}(\mathbf{W}) \doteq [\mathbf{W}_{\cdot 1}, \dots \mathbf{W}_{\cdot Q}]^\top$. Indeed, by construction $\langle \text{vect}(\mathbf{W}); \Psi(\boldsymbol{x}, r) \rangle = \langle \mathbf{W}_{\cdot r}; \boldsymbol{x} \rangle$. Moreover, this representation allows for the multiclass setting to be seen as a peculiar binary problem where the goal is to classify vectors of the form $[\Psi(\boldsymbol{x}, p) - \Psi(\boldsymbol{x}, q)]^\top$ into $+1$ and $-1$ whether $\langle \mathbf{W}_{\cdot p}; \boldsymbol{x} \rangle \geq \langle \mathbf{W}_{\cdot q}; \boldsymbol{x} \rangle$ or not.

Because we will not use this formalism in the present thesis though, we will not dwell further into the specifics of this approach; nonetheless interested readers may refer to, among others, [Crammer et al., 2006, Schapire and Singer, 2000] for further insights.

## 2.4 Conclusion

With this chapter, we conclude the first part of this thesis where we introduced the notions and ideas of what is considered today as the common ground of modern linear classification. Contrary to the previous chapter, we touched upon some more advanced machine learning's topics such as kernels and how the matters discussed in the previous chapter translate to this change of setting —that is, the necessity to drift from straight empirical risk minimization to structural risk minimization paradigm. Last but not least, we also briefly introduced the problem of multiclass classification as a direct extension to the binary case. Although multiclass classification is a relevant setting to this thesis as a whole —mostly as a natural extension of any bi-class method we may introduce— it will especially play a central role in Part II.

# Part II

# Learning With Noisy Labels

# Confusion Matrices for Multiclass Problems and Confusion Noise

3

Contents



Sometimes, the P.A.C.-setting, regardless of all its theoretical soudness, simply fails to capture the reality of a learning task. Mainly, this is because in practice the training set might not be trusted blindly when it comes to label reliability. Sometimes, obtaining labels directly from the target concept $t$ is too costly and one has to rely upon a rough estimate of $t$; sometimes, $t$ cannot be observed directly and any estimation of the value of $t$ is potentially corrupted by some external interferences. Regardless of the cause, this calls for learning models that account for label unreliability and are able to cope with it accordingly.

Such is the premise of Part II and the present chapter will introduce the relevant notions and works upon which chapter 4 is built. More precisely, we will assume that although some labels are corrupted —that is, erroneous— we have information on the underlying corruption pattern that we can leverage. The first part of this chapter accounts for previous works that have tackled the problem of corrupted labels —within the aforementioned limits— in the bi-class case; these works represent the fundamental ground for ours and were motivational in its conception. The

chapter's second part discusses the extension of this setting to multiclass classification through the introduction of *confusion matrices* and by doing so we will argue that multiclass problems call for errors and classifier disparity measures that are different to their bi-class counterparts.

## 3.1 A gentle introduction to noisy problems: the bi-class case

### 3.1.1 The general Agnostic setting

A very general extension of the regular P.A.C. setting (also referenced as *restricted* P.A.C. setting) is that of Agnostic P.A.C.learning [Kearns et al., 1994, Haussler, 1988], where the term agnostic comes from the idea that we assume to know nothing about the target concept $t$ except that it exists. Notably, it means that $t$ may potentially lie outside of $\mathbb{H}$. Formally, the goal of *Agnostic* P.A.C. learning is to find a classifier $h^* \in \mathbb{H}$ such that

$$R^{l_{0-1}}_{\mathscr{D}_t}(h^*) \doteq \min_{h \in \mathbb{H}} R^{l_{0-1}}_{\mathscr{D}_t}(h)$$

An interpretation of the Agnostic P.A.C. setting is that the labels observed in $\mathcal{S}$ are corrupted output from $h^*$, and learning under the P.A.C. setting amounts to learn $h^*$ from a corrupted dataset. It has been shown that, *Agnostic* P.A.C. learning encompasses all other noisy labels cases, including those when the corrupting process is malicious —or, in other words, when the labels are not corrupted independently from each others [Kearns et al., 1994]. The main drawback of having such a general setting is that *Agnostic* P.A.C. is notably difficult, and negative results abound in learning theory, e.g. [Feldman et al., 2012, Hfastad, 1997, Diakonikolas et al., 2011]. Interestingly though, the VC-dimension sample complexity bounds (theorem 1.2 for the regular P.A.C. setting hold in the agnostic case —minus some minor tweaks. An immediate consequence is that *Empirical Risk Minimization* is still a valid strategy in the agnostic P.A.C. setting. The true difficulty of this setting is actually to devise an algorithm that provably find $h^*$ with respect to the 0-1-loss, something known to be NP-hard [Höffgen et al., 1995, Johnson and Preparata, 1978, Angluin and Laird, 1987, Kearns and Li, 1993, Kearns et al., 1994], hence the relevance of surrogate approximate losses such as the *hinge loss* that are more easy to work with, but do not always match the solution defined by the 0-1-loss.

### 3.1.2 The Confusion Noise setting

With the *Agnostic* setting arguably too general for allowing interesting and practical learnability results, the tendency in the last years was to look for more flexible middle ground between *restricted* and *agnostic* P.A.C. [Bylander, 1994, Blum et al., 1998, Kalai et al., 2008]. In particular, an angle that will be relevant with the rest of this thesis consists in laying assumptions over the corrupting process —that is, how the labels are corrupted. Specifically, we are interested in the so-called *confusion noise* setting where the probability of switching a label to another is constant

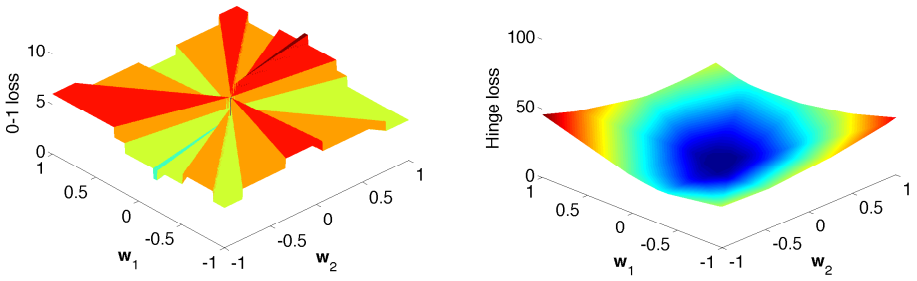



Figure 3.1 – *The 0-1 loss (left) and the hinge loss (right) in a similar fashion to figure 1.2 but on a noisy 2D dataset where no perfect classifier exists. Here, the 0-1 loss is clearly not convex contrary to the hinge loss. However, the two losses do not attain their minimal value at the same point.*

—as opposed to models where the noise depends on some localization properties. Namely, we write $\mathcal{S}_y \doteq \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ the corrupted dataset with $\mathcal{S}_y \overset{\text{i.i.d}}{\sim} \mathscr{D}_y^N$ where the $y_i$'s are corrupted versions of the *true labels* $t_i$'s. We call noise the process that turns $t$ into $y$ and we define the family of *confusion noises of rate* $\eta$, $0 \leq \eta < 0.5$ as the corrupting processes verifying:

$$\mathbb{P}\left[y(X) \neq t(X)\right] = \eta$$

While it is a seemingly easy setting to cope with, it has been shown that any algorithms that are *robust* to confusion noise —namely, any algorithms that can provably P.A.C. learn $t$ from a dataset corrupted with confusion noise— are also robust to *monotonic noises* [Bylander, 1994]. Where *monotonic noises* are classification noise processes such that the corruption rate of each point depends on its margin with $t$. Formally, let $\boldsymbol{w}^*$, $\|\boldsymbol{w}^*\|_2 = 1$ such that $h_{\boldsymbol{w}^*} \doteq t$, then monotonic noises follow the following property:

$$\mathbb{P}\left[t(X) \neq y(X)\right] = \left[1 - \frac{t(X)\left\langle \boldsymbol{w}^*; X\right\rangle}{\|X\|_2}\right]\eta$$

By construction, the true concept $t$ and its corrupted variant $y$ agree on a fraction $1-\eta$ of $\mathcal{S}_y$ and consequently, $t$ has an accuracy rate of $1-\eta$ over $\mathcal{S}_y$, something that other hypotheses in $\mathbb{H}$ may be able to achieve. Because of this, from a practical standpoint one cannot distinguish $t$ from any other hypothesis $h$ of accuracy rate $1-\eta$ without additional information. Hence, any hypothesis of accuracy $1-\eta$ is a valid solution.

Bearing this in mind, according to the P.A.C. paradigm, it is sufficient to solve this problem to devise an algorithm that provably find a hypothesis in $\mathbb{H}$ that achieve at least an accuracy rate of $1-\eta-\epsilon$ with probability $1-\delta$ over all possible instances of $\mathcal{S}$.

Two decades ago, [Bylander, 1994, Blum et al., 1998] both independently proposed a similar solutions to this problem. Their idea revolved around feeding a Perceptron algorithm with synthetic datapoints of known *true class* computed from the corrupted training set $\mathcal{S}$. In order to facilitate the notation, they introduces the so-called *normal form* of a bi-class training set.

**Definition 3.1** (Normal Form of a Bi-class Training Set) *Let $\mathcal{S} \doteq \{\boldsymbol{x}_i, t_i\}$ a bi-class training set with $t$ any concept (corrupted or not). We say that $\mathcal{S}'$ is the* normal form *of $\mathcal{S}$ if and only if*

$$\mathcal{S}' = \{(t_i \boldsymbol{x}_i, +1) | (\boldsymbol{x}_i, t_i) \in \mathcal{S}\}$$

To put it otherwise, datapoints in $\mathcal{S}$ with negative labels are reflected through $\mathbb{X}$'s origin. Notably, it means that for any point $\boldsymbol{x}$ in $\mathcal{S}$ and any classifier $\boldsymbol{w}$, there is a point $\boldsymbol{x}' \in \mathcal{S}'$ such that $\boldsymbol{x}' \doteq \boldsymbol{x}t(\boldsymbol{x})$, thus $t(\boldsymbol{x}) \langle \boldsymbol{w}; \boldsymbol{x} \rangle = \langle \boldsymbol{w}; \boldsymbol{x}' \rangle$ and learning from $\mathcal{S}'$ in our setting (that is, linear bi-class classification) is strictly equivalent to learning from $\mathcal{S}$. Henceforth, except stated otherwise, we will now assume that $\mathcal{S}$ is in normal form. Moreover, a key feature of the normal form is that all datapoints are now of class $(+1)$ and points with negative label correspond to prediction mistakes. Namely, given a noise rate $\eta$, it means that the learned classifier is allowed to predict of fraction $\eta$ of $\mathcal{S}_y$ in the negative class.

Lastly, from now on, we will distinguish the original dataset from the corrupted one by writing $\mathcal{S}_t$, as opposed to $\mathcal{S}_y$. Namely, $\mathcal{S}_t \overset{\text{i.i.d}}{\sim} \mathscr{D}_t^N$ and $\mathcal{S}_y \overset{\text{i.i.d}}{\sim} \mathscr{D}_y^N$.

### 3.1.3 An Algorithm for Learning Linear Classifier Under Classification Noise

We are now ready to describe the solution proposed by [Blum et al., 1998] for the problem of learning bi-class linear classifier under classification noise. We will focus on this solution rather than the one of [Bylander, 1994] as the two are very closely related. Although, the solution proposed in [Blum et al., 1998] is slightly more general as it does not forcefully require the existence of a large margin solution on the uncorrupted dataset $\mathcal{S}_t$.

The first step is independent of any noise process and consists in the introduction of a modified Perceptron algorithm (see the depiction in Alg. 3). Contrary to the usual Perceptron, Alg. 3 allows $\boldsymbol{w}$ to make mistakes on $\mathcal{S}_t$ as long as every misclassified points $\boldsymbol{x}$ are sufficiently close to the decision boundary. Namely: $|\langle \boldsymbol{w}; \boldsymbol{x} \rangle| \leq \sigma \|\boldsymbol{w}\|_2 \|\boldsymbol{x}\|_2$.

**Algorithm 3** [Blum et al., 1998] modified Perceptron

**Require:** A (uncorrupted) training set $\mathcal{S}_t \overset{\text{i.i.d}}{\sim} \mathscr{D}_t^N$ in normal form and a threshold parameter $\sigma$

1: Pick $\boldsymbol{w}^0$ at random in $\mathbb{X}$
2: $k \leftarrow 0$
3: **while** $\exists \boldsymbol{x} \in \mathcal{S}_t : \langle \boldsymbol{w}^k; \boldsymbol{x} \rangle < -\sigma \|\boldsymbol{w}^k\|_2 \|\boldsymbol{x}\|_2$ **do**
4: Pick any $\boldsymbol{x}_{\text{up}}^k \in \mathcal{S}_t$ such that $\left\langle \boldsymbol{w}^k; \boldsymbol{x}_{\text{up}}^k \right\rangle < -\sigma \|\boldsymbol{w}^k\|_2 \|\boldsymbol{x}_{\text{up}}^k\|_2$
5: $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k - \left\langle \boldsymbol{w}^k; \boldsymbol{x}_{\text{up}}^k \right\rangle \boldsymbol{x}_{\text{up}}^k$
6: $k \leftarrow k + 1$
7: **end while**
8: **return** $\boldsymbol{w}$

[Blum et al., 1998] state the following theorem on Alg. 3:

**Theorem 3.1** (Convergence of algorithm 3 [Blum et al., 1998]) *If $\mathcal{S}_t$ is labelled according to a linear concept $t$, then with probability $1-\delta$ Alg. 3 halts after $M$ iterations and outputs a vector $\boldsymbol{w}$ such that every misclassified point $\boldsymbol{x} \in \mathcal{S}_t$ satisfies $|\langle \boldsymbol{w}; \boldsymbol{x} \rangle| \leq \sigma \|\boldsymbol{w}\|_2 \|\boldsymbol{x}\|_2$ with*

$$M \in \mathcal{O}\left[\left(\frac{1}{\sigma^2}\right)\log(N)\log\left(\frac{1}{\delta}\right)\right]$$

Nonetheless, the authors make the following interesting remark about the proof. Let define $\boldsymbol{w}^*$, $\|\boldsymbol{w}^*\|_2 = 1$ such that $h_{\boldsymbol{w}^*} \doteq t$ then, for Theorem 3.1 to hold it suffices that the following conditions are met for $\boldsymbol{x}^k_{\text{up}}$ at any step $k$ (Alg. 3 line 4):

$$\left\langle \boldsymbol{w}^k; \boldsymbol{x}^k_{\text{up}} \right\rangle \leq \frac{-\sigma}{2\|\boldsymbol{w}^k\|_2 \|\boldsymbol{x}^k_{\text{up}}\|_2} \tag{3.1}$$

$$\left\langle \boldsymbol{w}^*; \boldsymbol{x}^k_{\text{up}} \right\rangle \geq \frac{-\sigma^2}{16\sqrt{N}\log N \|\boldsymbol{x}^k_{\text{up}}\|_2} \tag{3.2}$$

In plain English, it simply means that all update points must achieve a minimal, positive, margin with $\boldsymbol{w}^*$ (condition (3.2)) and, at the same time, $\boldsymbol{w}^k$ must errs on them with a large enough margin (condition (3.1)).

Building on this idea, [Blum et al., 1998] propose to cope with corrupted datasets by computing synthetic update vectors, as opposed to picking them directly in $\mathcal{S}_t$ which is not possible in a corrupted setting. In other words, because the algorithm has only access to $\mathcal{S}_y$ instead of $\mathcal{S}_t$, the conditions above are not verified on a fraction $\eta$ of the dataset —i.e. the corrupted examples. Thus, the idea is to manually build the needed update vectors from $\mathcal{S}_y$ in a way that ensures, with high probability, that those vectors fit the previously cited convergence conditions.

For a fixed step $k$, and any dataset $\mathcal{S}$, corrupted or not, let define:

$$\mathcal{S}^- \doteq \left\{ \boldsymbol{x} \in \mathcal{S} | \left\langle \boldsymbol{w}^k; \boldsymbol{x} \right\rangle < -\frac{\sigma}{\|\boldsymbol{w}^k\|_2 \|\boldsymbol{x}\|_2} \right\}$$
$$\mathcal{S}^+ \doteq \left\{ \boldsymbol{x} \in \mathcal{S} | \left\langle \boldsymbol{w}^k; \boldsymbol{x} \right\rangle > \frac{\sigma}{\|\boldsymbol{w}^k\|_2 \|\boldsymbol{x}\|_2} \right\}$$

By construction, we have that $\mathcal{S}_t^-$ contains only misclassified points that can potentially be used as update vectors —remember that $\mathcal{S}_t$ is in normal form. By linearity of the dot product, we have that $\sum_{\boldsymbol{x}\in\mathcal{S}_t^+} \boldsymbol{x}$ is also a valid update vector to be used in Alg. 3. Namely, let replace the selection of $\boldsymbol{x}_{\text{up}}$ (Alg. 3 line 4) by the following definition:

$$\boldsymbol{x}_{\text{up}} \doteq \sum_{\boldsymbol{x}\in\mathcal{S}_y^-} \boldsymbol{x} + \frac{\eta}{1-\eta} \sum_{\boldsymbol{x}\in\mathcal{S}_y^+} \boldsymbol{x} \tag{3.3}$$

The key observation here is that, with respect to the corruption process, one can rewrite $\sum_{\boldsymbol{x}\in\mathcal{S}_y^-} \boldsymbol{x}$ and $\sum_{\boldsymbol{x}\in\mathcal{S}_y^+} \boldsymbol{x}$ in terms of $\mathcal{S}_t^+$ and $\mathcal{S}_t^-$:

$$\mathbb{E}\left[\sum_{\boldsymbol{x}\in\mathcal{S}_y^-} \boldsymbol{x}\right] = (1-\eta)\sum_{\boldsymbol{x}\in\mathcal{S}_t^-} \boldsymbol{x} - \eta \sum_{\boldsymbol{x}\in\mathcal{S}_t^+} \boldsymbol{x} \tag{3.4}$$

$$\mathbb{E}\left[\sum_{\boldsymbol{x}\in\mathcal{S}_y^+} \boldsymbol{x}\right] = (1-\eta)\sum_{\boldsymbol{x}\in\mathcal{S}_t^+} \boldsymbol{x} - \eta \sum_{\boldsymbol{x}\in\mathcal{S}_t^-} \boldsymbol{x} \tag{3.5}$$

By re-injecting these two equations into equation (3.3) we ultimately have:

$$\mathbb{E}\left[\boldsymbol{x}_{\text{up}}\right] = \left(\frac{1-2\eta}{1-\eta}\right) \sum_{\boldsymbol{x} \in \mathcal{S}_t^-} \boldsymbol{x}$$

Thus, the expected value of $\boldsymbol{x}_{\text{up}}$ acts as a proxy of $\mathcal{S}_t^-$ and therefore $\boldsymbol{x}_{\text{up}}$ is a valid update vector. Namely, running algorithm 3 on $\mathcal{S}_y$ with $\boldsymbol{x}_{\text{up}}$ defined as above amounts to, expectation wise, directly run Alg. 3 on $\mathcal{S}_t$ with picking $\boldsymbol{x}_{\text{up}}$ as the sum of the misclassified point in $\mathcal{S}_t$.

Last but not least, it remains to tackle the matter of how far $\boldsymbol{x}_{\text{up}}$ actually is from its expected value $\mathbb{E}\left[\boldsymbol{x}_{\text{up}}\right]$. In their work, [Blum et al., 1998] show that as long as $|\mathcal{S}_t^+ \cup \mathcal{S}_t^-|$ is large enough —note that $|\mathcal{S}_t^+ \cup \mathcal{S}_t^-| \leq |\mathcal{S}_t|$— then either $|\boldsymbol{x}_{\text{up}} - \mathbb{E}\left[\boldsymbol{x}_{\text{up}}\right]|$ is small enough for conditions (3.1) and (3.2) to hold or, otherwise, $\boldsymbol{w}^k$ is already a good enough classifier and the algorithm then halts.

This conclude the analysis of [Blum et al., 1998] which will be our starting point when dealing with multiclass classification. However, before going there, we will have to properly extend the various definition and notions seen so far to the case of multiple output classes, which is what the remaining of this chapter is devoted to.

## 3.2 Confusion Matrices

### 3.2.1 A note on Precision and Recall

Before venturing to the topic of confusion matrices for multiclass problems, a good preliminary is to discuss the case of precision/recall matrices for bi-class problems.

Until now, we have defined the *risk* independently of the positive and negative classes. That is to say, we always have assumed the *risk* to be some value $\eta$ holding over the totality of $\mathcal{S}$ in a way that, for a given classifier $h$:

$$\mathbb{P}_{X \sim \mathscr{D}}\left[h(X) \neq t(X)\right]$$

In most cases this definition of risk is adequate, both practically and theoretically, but sometimes problems arise when one need a finer level of granularity in performance measurement.

**Example 3.1** (Toxic Mushroom) *Let consider the task of predicting the toxicity of mushroom species from biological data. It is well established that some cases of mushroom poisoning can be very dire, or even deadly. Typically, erroneously predicting a mushroom to be safe can lead to threatening human lives while, on the other hand, the converse —that is, erroneously predicting a mushroom as toxic— has little to no consequences. This is precisely the kind of problems where we want to minimize false negative cases —that is, mushroom erroneously predicted safe— at all cost, even if it means an increase of false positive cases. Because the true risk, as defined previously, is impervious to these subtlety, we thus need a better performance measure to assert and constrain the quality of the produced classifiers*

Building on this example, for a given classifier $h$ we distinguish multiple prediction scenarii based on the respective output of $t$ and $h$:

**Definition 3.2** (The different types of error) *For a given concept $t$ and classifier $h$ we distinguish the following outcomes:*

- *True Positive:* $t = +1$ *and* $h = +1$
- *False Positive:* $t = -1$ *and* $h = +1$
- *False Negative:* $t = +1$ *and* $h = -1$
- *True Negative:* $t = -1$ *and* $h = -1$

Alternatively, it is common to refer to *false positive* outcomes as *Type-1* errors and *false negatives* as *Type-2* errors. Hence, the total number of errors —that is, the measure we used to work with— is simply the sum of the *Type-1* and *Type-2* errors. These definitions in hand, we then introduce the corresponding probability rates. Namely:

**Definition 3.3** (The different probability error rates) *For a given concept $t$ and classifier $h$ we defines the following probability rates:*

- *True Positive Rate (TPR):*

$$\mathbb{P}\left[h(X) = +1 | t(X) = +1\right] \doteq \frac{\mathbb{E}\left[\mathbb{I}\left[h(X) = +1 \wedge t(X) = +1\right]\right]}{\mathbb{E}\left[\mathbb{I}\left[t(X) = +1\right]\right]}$$

- *False Positive Rate (FPR):*

$$\mathbb{P}\left[h(X) = +1 | t(X) = -1\right] \doteq \frac{\mathbb{E}\left[\mathbb{I}\left[h(X) = +1 \wedge t(X) = -1\right]\right]}{\mathbb{E}\left[\mathbb{I}\left[t(X) = -1\right]\right]}$$

- *False Negative Rate (FNR):*

$$\mathbb{P}\left[h(X) = -1 | t(X) = +1\right] \doteq \frac{\mathbb{E}\left[\mathbb{I}\left[h(X) = -1 \wedge t(X) = +1\right]\right]}{\mathbb{E}\left[\mathbb{I}\left[t(X) = +1\right]\right]}$$

- *True Negative Rate (TNR):*

$$\mathbb{P}\left[h(X) = -1 | t(X) = -1\right] \doteq \frac{\mathbb{E}\left[\mathbb{I}\left[h(X) = -1 \wedge t(X) = -1\right]\right]}{\mathbb{E}\left[\mathbb{I}\left[t(X) = -1\right]\right]}$$

*Where all probabilities and expectations are conditioned over $X \overset{i.i.d}{\sim} \mathscr{D}_t$ and $\mathbb{I}\left[\cdot\right]$ is the indicator function that return $+1$ if its argument evaluates to* true *and* $0$ *otherwise.*

Note that, while the number of errors (resp. number of good predictions) can be obtained by summing *false positive* and *false negative* outcomes (resp. *true positive* and *true negative*) the *risk* (resp. *accuracy*) cannot be computed from FPR and FNR (resp. TPR and TNR) alone. Indeed, by using probability rates conditioned over the value of $t(X)$ we negate the effect of class repartition. In other words, to retrieve the *risk* (resp. *accuracy*) one have to multiply the FNR (resp. TPR) and FPR (resp. TNR) by the corresponding class weight in $\mathcal{S}$, that is $\mathbb{P}\left[t(X) = +1\right]$ and $\mathbb{P}\left[t(X) = -1\right]$, then add the two resulting values together.

$$R^{l_{0-1}}_{\mathscr{D}_t}(h) = \text{FNR} \times \mathbb{P}\left[t(X) = +1\right] + \text{FPR} \times \mathbb{P}\left[t(X) = -1\right]$$

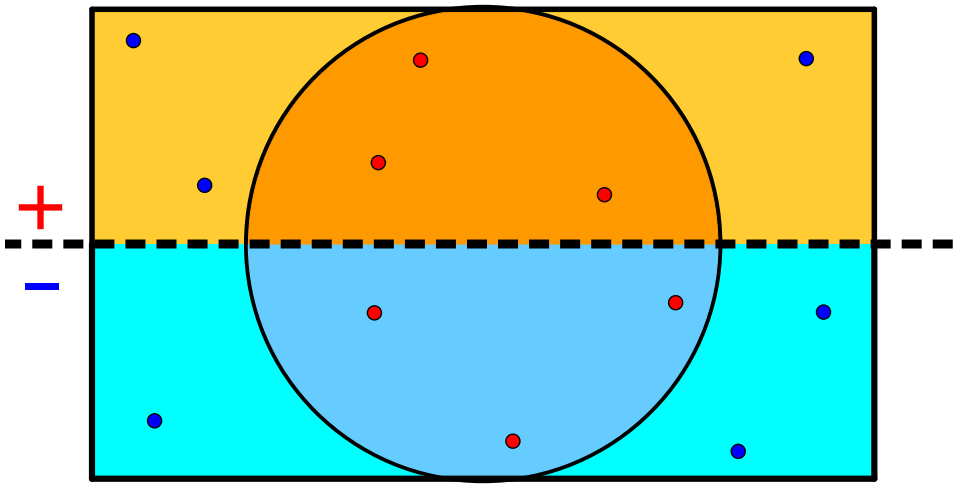



Figure 3.2 – *A schematic representation of the different types of error. In this example the target concept $t$ is modelled by the black circle and the data are labelled positive (red) or negative (blue) whether they lie within the circle. The current hypothesis is represented by the dashed black line and predict the data above (resp. below) the line as positive (resp. negative). The four areas correspond to the four possible prediction outcomes: True Positive (orange), True Negative (cyan), False Positive (yellow) and False Negative (blue). By counting how many points fall within each region, it is possible to estimate the error rates of definition 3.3 For instance, the* precision *is defined as the number of True Positive (i.e. the number of points in the orange region) divided by the number of Positive examples (the orange and blue regions).*

The obvious upside of those alternative error rates is that they are completely impervious of class imbalance —that is, situations where one class is vastly supernumerary comparatively to the other one.

Other possible measures related to these definitions include, among others, the *Positive Prediction Value (PPV)*, *False Discovery Rate (FDR)*, *False Omission Rate (FOR)* and *Negative Predictive Value (NPV)* which are used and studied by the field of ROC analysis. The interested reader can find proper definitions and extended details for those measure in some introductory works related to ROC analysis such as [Fawcett, 2006]. Also, note that *precision* and *recall* are sometimes used as an alternative to this set of measures. However, *precision* is actually an other name for PPV and *recall* corresponds to TPR; as such TP/FP/FN/TN rates and *precision*/*recall* are generally not used together. More precisely, we can define precision as $\mathbb{P}(t(X) = +1 | h(X) = +1)$ and recall as $\mathbb{P}(h(X) = +1 | t(X) = +1)$

A synthetic way to manipulate the previously defined probability rates —that is, TPR, FPR, FNR and TNR— is through a *left stochastic matrix* of the form

$$\mathbf{C} \doteq \begin{pmatrix} \text{TP rate} & \text{FP rate} \\ \text{FN rate} & \text{TN rate} \end{pmatrix}$$

Or, in a more formal way, $\mathbf{C} \in \mathbb{R}^{2\times 2}$ is the matrix

$$\mathbf{C} \doteq \begin{pmatrix} \mathbb{P}(h(X) = +1 | t(X) = +1) & \mathbb{P}(h(X) = +1 | t(X) = -1) \\ \mathbb{P}(h(X) = -1 | t(X) = +1) & \mathbb{P}(h(X) = -1 | t(X) = -1) \end{pmatrix} \tag{3.6}$$

where the dependency on $h$ is made implicit in an attempt to keep the notations clear.

Moreover, note that we have for any row $j$

$$\mathbf{C}_{1j} + \mathbf{C}_{2j} = 1$$

because, $\mathbf{C}$ is a *left stochastic matrix* and thus $\mathbf{C}_{1j} = 1 - \mathbf{C}_{2j}$. In addition, row-wise, remark that we have:

$$\begin{aligned}\mathbf{C}_{11} + \mathbf{C}_{12} &= \mathbb{P}(h(X) = +1)\\ \mathbf{C}_{21} + \mathbf{C}_{22} &= \mathbb{P}(h(X) = -1)\end{aligned}$$

Finally, the diagonal terms $\mathbf{C}_{11}$ and $\mathbf{C}_{22}$ are related to the accuracy as follow

$$\mathbb{P}(h(X) = t(X)) = \mathbf{C}_{11}\mathbb{P}(t(X) = 1) + \mathbf{C}_{22}\mathbb{P}(t(X) = -1)$$

and the off-diagonal terms $\mathbf{C}_{12}$ and $\mathbf{C}_{21}$ to the risk:

$$\begin{aligned}R^{l_{0-1}}_{\mathscr{D}_t}(h) &= \mathbf{C}_{12}\mathbb{P}(t(X) = -1) + \mathbf{C}_{21}\mathbb{P}(t(X) = 1)\\ &= (1 - \mathbf{C}_{22})\mathbb{P}(t(X) = -1) + (1 - \mathbf{C}_{11})\mathbb{P}(t(X) = 1)\\ &= \mathbb{P}(t(X) = -1) + \mathbb{P}(t(X) = +1)\\ &\quad - [\mathbf{C}_{22}\mathbb{P}(t(X) = -1) + \mathbf{C}_{11}\mathbb{P}(t(X) = 1)]\\ &= 1 - \mathbb{P}(h(X) = t(X))\end{aligned}$$

Where the last line come from the fact that $\mathbb{P}(t(X) = -1) + \mathbb{P}(t(X) = +1) = 1$.

In the past years, there have been some works that have explored the use of these notions as performance measures for linear classifier, or some similar ideas. Among them, we may cite [Li et al., 2005] that copes with the problem of unbalanced dataset where one class is a lot more numerous than the other.

Now that we have introduced these notions for bi-class settings, we are ready to generalize them to multiple classes problems and properly introduce confusion matrices.

### 3.2.2 Confusion Matrices for Multiclass Problems

In this section, we formally introduce *confusion matrices*. That is, multi-class extension of the previously defined bi-class matrix $\mathbf{C}$. Namely, for a multiclass problem of $Q$ classes, we extend the previous definition of $\mathbf{C}$ by building upon the general term definition given in equation (3.6) and we define, for a given concept $t$ and classifier $h$, the confusion matrix of $h$, $\mathbf{C} \in \mathbb{R}^{Q\times Q}$ of general term

$$\mathbf{C}_{ij} \doteq \mathbb{P}(h(X) = i | t(X) = j)$$

where, as usual and unless stated otherwise, probabilities are conditioned over $X \overset{\text{i.i.d}}{\sim} \mathscr{D}_t$. Hence, $\mathbf{C}$ has the following general structure:

$$\mathbf{C} \doteq \begin{pmatrix} \mathbb{P}(h(X)=1|t(X)=1) & \cdots & \mathbb{P}(h(X)=1|t(X)=Q) \\ \mathbb{P}(h(X)=2|t(X)=1) & \cdots & \mathbb{P}(h(X)=2|t(X)=Q) \\ \vdots & \ddots & \vdots \\ \mathbb{P}(h(X)=Q|t(X)=1) & \cdots & \mathbb{P}(h(X)=Q|t(X)=Q) \end{pmatrix}$$

and similarly to the bi-class case, we have that $\mathbf{C}$ is a *left stochastic matrix* with

$$\begin{aligned} \forall i \in [Q] : \sum_{j\in[Q]} \mathbf{C}_{ij} &= \sum_{j\in[Q]} \mathbb{P}(h(X)=i|t(X)=j) \\ &= \mathbb{P}(h(X)=i) \end{aligned}$$

$$\begin{aligned} \forall j \in [Q] : \sum_{\substack{i\in[Q] \\ i\neq j}} \mathbf{C}_{ij} &= \sum_{\substack{i\in[Q] \\ i\neq j}} \mathbb{P}(h(X)=i|t(X)=j) \\ &= \mathbb{P}(h(X)\neq j) \\ &= 1-\mathbf{C}_{jj} \end{aligned}$$

$$\begin{aligned} \sum_{i\in[Q]} \mathbf{C}_{ii}\mathbb{P}(t(X)=i) &= \sum_{i\in[Q]} \mathbb{P}(h(X)=i \wedge t(X)=i) \\ &= \mathbb{P}(h(X)=t(X)) \\ &= 1-R^{l_{0-1}}_{\mathscr{D}_t}(h) \end{aligned}$$

In substance, confusion matrices contain the errors rate of a classifier for any pair of predicted/true classes, as such they provide a class-wise information about the error patterns of a classifier. Although such level of detail is situational at best in bi-class problems, it is an altogether more important matter for multiclass ones. Arguably, on a conceptual level, multiclass classification is about capturing the essence of each class, therefore the performance measure — whether it is the risk, or anything else— act as a numerical proxy that asserts the success of this primal goal. The risk is a good performance measure in this regards for bi-class classification because of the limited output possibilities (either $+1$ or $-1$). On the other hand, in a multiclass setting, using such a global measure means that improving accuracy over one class can compensate errors in another one. In particular, when classes are imbalanced, it means that improving accuracy by, say, 10% over given class with a lot of example can allows to completely ignore other classes with few datapoints. Typically, this is something that should not be allowed given our original goal of correctly learning the nature of all classes. In this regard, confusion matrices are a tool that allows much finer control over how the errors are spread among the different classes of the dataset. However, the question that remains is the one of exploiting the information contained in confusion matrices in a way that is relevant to the goal of learning a collection of $Q$ classes; and to this question, a definite answer has yet to be given, if any exists.

Recently, the idea of using confusion matrices as performance estimators gained momentum though the publication of positive theoretical and

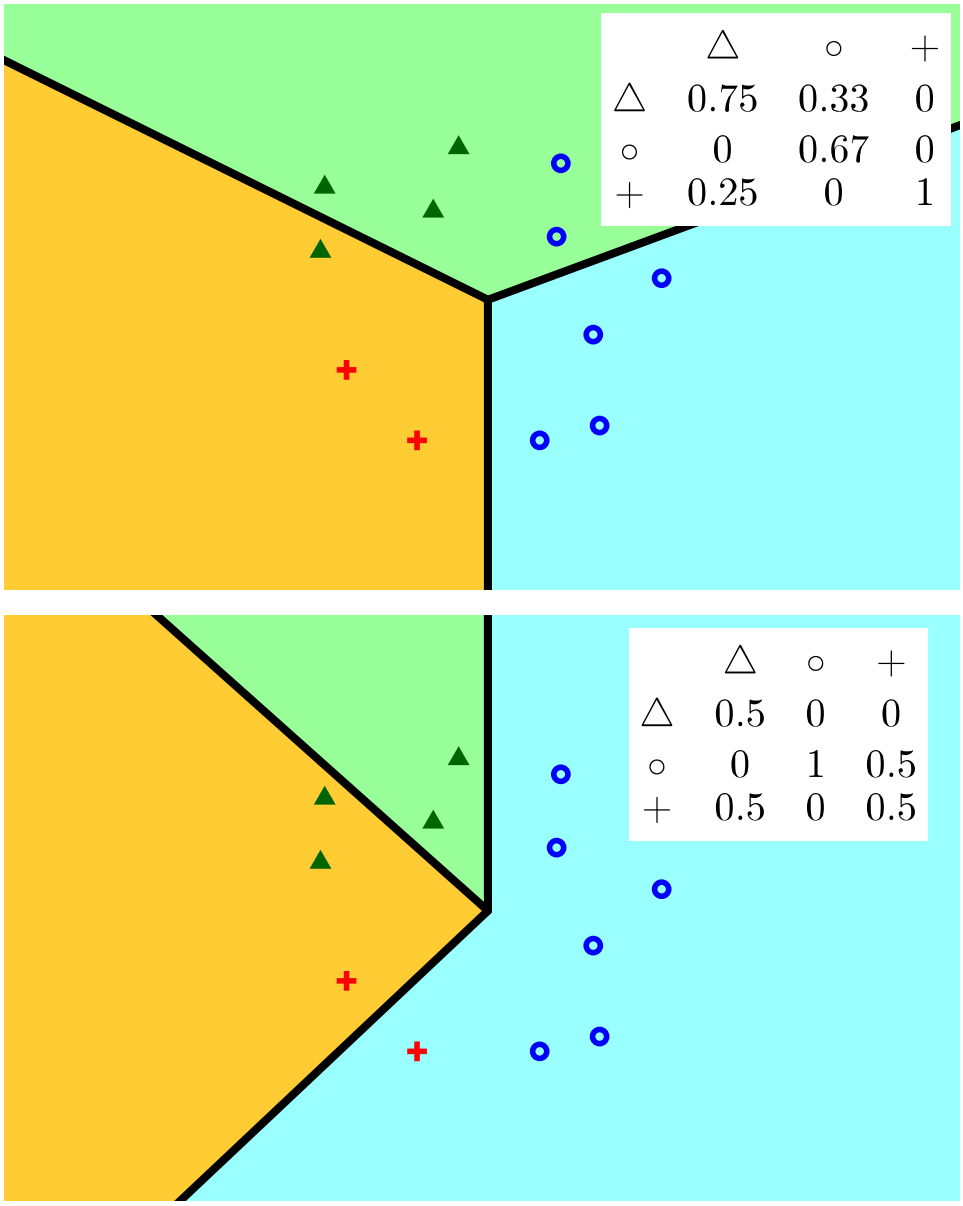



Figure 3.3 – *An illustration of two different linear classifiers on the same dataset. The data are spread over* 3 *classes* (△, ○ *and* +). *The points located in the green (resp. blue and orange) region are predicted of class* △ *(resp.* ○, +). *In both figures the prediction error rates have the same value of* 0.25 *yet the corresponding confusion matrices (top-right corner) are different. The Frobenius norm (see definition ADDREF) of the confusion risk is* 0.17 *in the top figure but* 0.50 *in the bottom one. Indeed, we can observe that the top classifier makes some mistakes on the most represented class* (○) *whereas the bottom classifier is wrong on half the points for two classes out of three.*

practical results. These works consider the *confusion risk* which is simply the confusion matrix without its diagonal:

$$R^{\mathbf{C}}_{\mathscr{D}_t}(h) \doteq \begin{cases} \mathbf{C}_{ij} & \text{if} \quad i \neq j \\ 0 & \text{otherwise} \end{cases}$$

where we overloaded the risk notation in the above definition.

The majority of the works on *confusion risk* have focused on minimizing the *operator norm* of $R^{\mathbf{C}}_{\mathscr{D}_t}(h)$ for algebraic reasons, which is simply defined, for any given matrix $M$ as

$$\|M\|_{\text{op}} \doteq \max_{v \neq \mathbf{0}} \frac{\|M\boldsymbol{v}\|_2}{\|\boldsymbol{v}\|_2}$$

The first theoretical results on the subject is due to [Machart and Ralaivola, 2012] which have established algorithmic stability bounds for confusion matrices in the addition to showing that some existing algorithm already exhibit some confusion stability properties. A second theoretical results was proposed later by [Morvant et al., 2012] that established P.A.C.-Bayes bounds for confusion matrices. Finally, quickly after, [Ralaivola, 2012] successfully proposed a new algorithm, named COPA for *Confusion-based Online Passive Aggressive*, that was based on a variation of the ultraconservative additive update scheme (see [Crammer et al., 2006]) but was designed to minimize the *operator norm* of $R^{\mathbf{C}}_{\mathscr{D}_t}(h)$. Notably experimental results for COPA backed the emerging confusion risk theory, with state-of-the-art accuracy rates but combined to a much lower confusion risk. In other words, the errors made by COPA, while being globally as numerous as with other methods, were more evenly distributed among the classes, thus leading to an overall better portrayal of the different $Q$ classes.

## 3.3 The Multiclass Confusion Noise Model

In this work though, we will not be using confusion matrices as performance measure. Nonetheless, the previous preparatory allowed us to introduce confusion matrices in a somewhat more natural way and provide a prime example of how they can be relevant to multiclass settings. Notably, we have discussed how multiclass problems call for finer, class-wise, analytic tools. This is somewhat in opposition to the usual idea that multiclass problems can be emulated, both theoretically and algorithmically by a system of bi-class problems. In particular, ideas that are often overlooked in bi-class problems are central when dealing with multiple classes. In this last section, we will describe how confusion matrices naturally extends the notion of corrupted dataset previously discussed in section 3.1

Namely, we are interested in noisy multiclass settings, where the noise pattern is a multiclass extension of the bi-class noise. To this end, we establish a proper definition of multiclass confusion noise.

We start from the previous idea of confusion matrices as characterizations of the error patterns of a classifier $h$. Except that we will be interested in the differences between the true, pristine, concept $t$ and its corrupted

variant $y$. Hence, the confusion matrix we will be interested in is associated with $y$ and can be alternatively thought as a class-wise description of the noising process that corrupt $\mathcal{S}_y$.

More precisely, for a noisy multiclass problem with $Q$ classes, we define the confusion matrix $\mathbf{C}$ as the matrix of dimension $Q \times Q$ (i.e. $\mathbf{C} \in \mathbb{R}^{Q \times Q}$) of general term

$$\mathbf{C}_{ij} \doteq \mathbb{P}(y(X) = i | t(X) = j)$$

Hence, we introduce *multiclass confusion noises* as the noise processes that can be fully described by a confusion matrix. Notably, this implies that for two given classes $p$ and $q$ the probability of observing the label $p$ instead of the label $q$ —where $p$ is a corrupted label and $q$ the true class— is constant for all $\boldsymbol{x} \in \mathbb{X}$ of class $q$ and equal to $\mathbf{C}_{pq}$.

Alternatively put, the noise process is *uniform* within each class and its level does not depend on the precise location of $\boldsymbol{x}$ within the region that corresponds to the class $t(\boldsymbol{x})$. As such, this noise process is both a) very aggressive, as it does not only apply to regions close to the class boundaries between classes and b) regular, in the sense that the mislabelling rate is piecewise constant. Nonetheless, this setting can account for many real-world problems as numerous noisy phenomena can be summarized by a simple confusion matrix. Moreover it has been proved [Bylander, 1994] that robustness to classification noise generalizes robustness to monotonic noise where, for each class, the noise rate is a monotonically decreasing function of the distance to the class boundaries.

**Remark 3.1** *The confusion matrix $\mathbf{C}$ should not be mistaken with the matrix $\tilde{\mathbf{C}}$ of general term: $\tilde{\mathbf{C}}_{ij} \doteq \mathbb{P}(t(X) = i | y(X) = j)$ which is the class-conditional distribution of $t(X)$ given $y$. The problem of learning from a noisy training set and $\tilde{\mathbf{C}}$ is a different problem than the one we aim to solve. In particular, $\tilde{\mathbf{C}}$ can be used to define cost-sensitive losses rather directly whereas doing so with $\mathbf{C}$ is far less obvious. Anyhow, this second problem of learning from $\tilde{\mathbf{C}}$ is far from trivial and very interesting but ultimately falls beyond the scope of the present work.*

Also, note that this setting trivially encompasses other noise framework where the noise is described on a more general scale than pair-wise classes combination. For instance, the noise process that, for any point $\boldsymbol{x} \in \mathbb{X}$ of any class, randomly switch its class with probability $\eta$ into any other classes can be described in our setting by a confusion matrix with $(1 - \eta)$ on all its diagonal values and $\eta / (Q - 1)$ everywhere else.

Obviously, all of the previous properties of confusion matrices (see section 3.2) hold in this setting. Notably, $\mathbf{C}$ is a *left-stochastic* matrix and, consequently, one can retrieve the probability distribution of the corrupted classes from $\mathbf{C}$ and the true class distribution —however, in the general case, this distribution is unknown. In other words, let define $\boldsymbol{\pi}_t$ (resp. $\boldsymbol{\pi}_y$) the probability vector of general term $(\boldsymbol{\pi}_t)_i \doteq \mathbb{P}(t(X) = i)$ (resp.

$(\boldsymbol{\pi}_y)_i \doteq \mathbb{P}(y(X) = i)$), then the following equality holds:

$$\begin{aligned}\forall i \in [Q] : \sum_{j=1}^{Q} \mathbf{C}_{ij} (\boldsymbol{\pi}_t)_j &= \sum_{j=1}^{Q} \mathbb{P}(y(X) = i | t(X) = j)\mathbb{P}(t(X) = j) \\ &= \mathbb{P}(y(X) = i) \\ &= (\boldsymbol{\pi}_y)_i\end{aligned}$$

Alternatively, in matrix form:

$$\mathbf{C} \times \boldsymbol{\pi}_t = \boldsymbol{\pi}_y$$

Particularly, from algebra, we have that if $\mathbf{C}$ is invertible, there is a closed form for $\boldsymbol{\pi}_t$ depending only in $\mathbf{C}$ and $\boldsymbol{\pi}_y$:

$$\boldsymbol{\pi}_t = \mathbf{C}^{-1} \times \boldsymbol{\pi}_y \tag{3.7}$$

Notably, this trick is one of the core idea that is developed in Chapter 4 and will allow learning in the context of multiclass confusion noise. In particular, a prominent question that will be tackled is how to handle empirical estimation of these probabilistic values. Also, note that equation (3.7) is a generalization of the relation between $\mathcal{S}_t^+$, $\mathcal{S}_t^-$, $\mathcal{S}_y^+$ and $\mathcal{S}_y^-$ presented in section 3.1. Namely, if $Q = 2$, equation (3.7) roughly reduces to equation (3.4) and equation (3.5).

## 3.4 Conclusion

In this chapter we have presented two of the premises of the work we will expose next. In the first section, we have discussed the matter of learning from noisy datasets in the setting of bi-class classification and we have introduced some positive results and algorithms relative to this setting. Additionally, the second section revolved around multiclass notions, and how *confusion matrices* play a major role into asserting the quality of a classifier in a multiclass setting. Moreover, the second part's discussion conveys the message that, contrary to what it may look like, multiclass classification problems should not be thought as trivial extensions of bi-class ones, especially when it comes to error measurement.

Ultimately, we will combine these two ideas in the next chapter and tackle the problem of learning under confusion noise in a multiclass setting. As such, this chapter was a mean to properly introduce the background knowledge relevant to chapter 4.

# Unconfused Multiclass Algorithms

4

Contents



This chapter deals with multiclass linear classification problems with confused (corrupted) data defined as in section 2.3 and 3.3.

Our main contribution is to show that it is both practically and theoretically possible to learn a multiclass classifier on noisy data if some information on the noise process is available. We propose a way to generate new points for which the true class is known. Hence we can iteratively populate a new *unconfused* dataset to learn from. This allows us to handle a massive amount of mislabelled data with only a very slight loss of accuracy. We embed our method into ultraconservative algorithms and provide a thorough analysis of it, in which we show that

the strong theoretical guarantees that characterize the family of ultraconservative algorithms carry over to the noisy scenario. The resulting algorithm, that we call UMA —for *Unconfused Multiclass Additive* algorithm— is, to the best of our knowledge, the first method to make use of the confusion matrix as a way to handle corrupted data in a multiclass setting. The present work has given birth to two publications, one in the *Asian Conference on Machine Learning* in 2013 [Louche and Ralaivola, 2013] and the second as an extended version in *Machine learning Journal* in 2015 [Louche and Ralaivola, 2015b]. The exposition of this chapter will closely follow that of the aforementioned papers. Namely, Section 4.1 will discuss the specifics and assumptions of our setting, we will expound UMA in section 4.2 and provide a detailed analysis of the algorithm, section 4.3 is devoted to experimental simulations and lastly section 4.4 will wrap up this chapter with a more general discussion of this work.

## 4.1 Setting and Problem

### 4.1.1 A gentle start and a practical example

The probabilistic setting we consider hinges on the existence of two components. On one hand, we assume the usual unknown (but fixed) probability distribution $\mathscr{D}$ on $\mathbb{X} \doteq \mathbb{R}^D$. On the other hand, we also assume the existence of a deterministic labelling function $t \in \mathbb{H} \subset \mathbb{Y}^{\mathbb{X}}$, where $\mathbb{Y} \doteq \{1, \dots Q\}$, which associates a label $t(\boldsymbol{x})$ to any input example $\boldsymbol{x}$. In particular, we are interested in establishing the robustness of *ultraconservative additive* algorithms (see Alg. 2) to label confusion noise in the multiclass setting. As we want to recover from the confusion noise, *i.e.*, we want to achieve low risk on uncorrupted/non-noisy data, we use the term *unconfused* to characterize the procedures we propose.

Beside the theoretical questions raised by the learning setting considered, we may depict the following example of an actual learning scenario where learning from noisy data is relevant. This learning scenario will be further investigated from an empirical standpoint in the section devoted to numerical simulations (Section 4.3).

**Example 4.1** *One situation where coping with mislabelled data is required arises in (partially supervised) scenarios where labelling data is very expensive. Imagine a task of text categorization from a training set* $\mathcal{S} \doteq \mathcal{S}_{base} \cup \mathcal{S}_{missing}$*, where* $\mathcal{S}_{base} = \{(\boldsymbol{x}_i, t_i)\}_{i=1}^{N}$ *is a set of N labelled training examples and* $\mathcal{S}_{missing} = \{\boldsymbol{x}_{N+i}\}_{i=1}^{M}$ *is a set of M unlabelled vectors; in order to fall back to realistic training scenarios where more labelled data cannot be acquired, we may assume that* $N \ll M$*. A possible three-stage strategy to learn a predictor is as follows: first learn a predictor* $h_{base}$ *on* $\mathcal{S}_{base}$ *and estimate its confusion error* $\mathbf{C}$ via *a cross-validation procedure —*$h_{base}$ *is assumed to make mistakes evenly over the class regions— second, use the learned predictor to label all the data in* $\mathcal{S}_{missing}$ *to produce the labelled training set* $\widehat{\mathcal{S}} \doteq \mathcal{S}_{base} \cup \{(\boldsymbol{x}_{N+i}, h_{base}(\boldsymbol{x}_{N+i}))\}_{i=1}^{M}$ *and finally, learn a classifier h from* $\widehat{\mathcal{S}}$ and *the confusion information* $\mathbf{C}$*.*

This introductory example pertains to semi-supervised learning and this is only one possible learning scenario where the contribution we pro-

pose, UMA, might be of some use. Still, it is essential to understand right away that one key feature of UMA, which sets it apart from many contributions encountered in the realm of semi-supervised learning, is that we do provide theoretical bounds on the sample complexity and running time required by our algorithm to output an effective predictor.

### 4.1.2 Assumption

Out setting is built on a set of assumptions that will hold throughout this chapter. In order to ease the comprehension, we will state them all at once in this section before moving on the exposition of our work.

First, without loss of generality, we will suppose that the data are normalized over the sphere of unit norm:

$$\mathbb{P}\left(\|X\|_2 = 1\right) = 1$$

Additionally, because we are interested in learning a linear classifier we shall consider the linear separability assumption of theorem 2.3 to hold with margin $\gamma_{\mathcal{S}_t}^{\mathbf{W}^*}$ on $\mathcal{S}_t$. Namely

**Assumption 4.1** (Linear Separability of $\mathcal{S}_t$ with Margin $\gamma$.) *There exist $\gamma > 0$ and $\mathbf{W}^* = [\mathbf{W}^*_{\cdot 1} \cdots \mathbf{W}^*_{\cdot Q}] \in \mathbb{R}^{D\times Q}$, with $\|\mathbf{W}^*\|_F^2 = 1$ (see def 2.4) such that $h_{\mathbf{W}^*}$ has margin $\gamma_{\mathcal{S}_t}^{\mathbf{W}^*} \geq \gamma$.*

As announced in the chapter introduction, we follow the *multiclass confusion noise* setting described in chapter 3, section 3.3 and assume that we only have access to a corrupted version

$$\mathcal{S}_y \doteq \{(\boldsymbol{x}_i, y_i))\}_{i=1}^N \tag{4.1}$$

where, as before, each $y_i$ is a short hand to $y(\boldsymbol{x}_i)$ with $y \in \mathbb{Y}^{\mathbb{X}}$ a corrupted version of the *true concept* $t$ that complies to the following assumption:

**Assumption 4.2** (Confusion Noising Process) *The corrupted concept $y$ —alternatively, the law $\mathscr{D}_y$ of $(X, Y)$— is so that the conditional distribution*

$$\mathbb{P}_{(X,Y)\sim\mathscr{D}_y}(Y|t(X))$$

*is fully summarized into a* known *confusion matrix $\mathbf{C} \in \mathbb{R}^{Q\times Q}$ given by*

$$\begin{aligned}\forall \boldsymbol{x} \in \mathbb{X} : \mathbf{C}_{pq} &\doteq \mathbb{P}_{(X,Y)\sim\mathscr{D}_y}(Y = p|t(X) = q)\\ &\doteq \mathbb{P}_{X\sim\mathscr{D}}(y(X) = p|t(X) = q)\end{aligned} \tag{4.2}$$

Finally, we assume the following from here on:

**Assumption 4.3** ($\mathbf{C}$ Invertibility) $\mathbf{C}$ *is invertible.*

Note that this assumption is not as restrictive as it may appear. For instance, if we consider the learning setting depicted in Example 4.1 and implemented in the numerical simulations, then the confusion matrix obtained from the first predictor $h_{\text{base}}$ is often diagonally dominant, i.e. the magnitudes of the diagonal entries are larger than the sum of the magnitudes of the entries in their corresponding rows, and $\mathbf{C}$ is therefore invertible. Generally speaking, the problems that we are interested in (i.e.

problems where the true classes seems to be recoverable) tend to have invertible confusion matrix. It is most likely that invertibility is merely a sufficient condition on $\mathbf{C}$ that allows us to establish learnability in the sequel. Identifying less stringent conditions on $\mathbf{C}$, or conditions termed in a different way —which would for instance be based on the condition number of $\mathbf{C}$— for learnability to remain, is a research issue of its own that we leave for future investigations.

### 4.1.3 Problem: Learning a Linear Classifier from Noisy Data

The problem we address is thus the learning of a classifier $h$ from $\mathcal{S}_y$ *and* $\mathbf{C}$ so that the error rate

$$R^{l_{0-1}}_{\mathscr{D}_t}(h) = \mathbb{P}(h(X) \neq t(X))$$

of $h$, is as small as possible: the usual goal of learning a classifier $h$ with small risk is preserved, while now the training data is only made of corrupted labelled pairs.

Building on Assumption 4.1, we may refine our learning objective by restricting ourselves to linear classifiers $h_{\mathbf{W}}$, for $\mathbf{W} = [\mathbf{W}_{\cdot 1} \cdots \mathbf{W}_{\cdot Q}] \in \mathbb{R}^{D \times Q}$ (see Definition 2.3). Our goal is thus to learn a relevant matrix $\mathbf{W}$ from $\mathcal{S}_y$ *and* the confusion matrix $\mathbf{C}$. More precisely, we achieve risk minimization through classic additive methods and the core of this chapter is focused on computing noise-free update points such that the properties of said methods are unchanged.

## 4.2 UMA: Unconfused Ultraconservative Multiclass Algorithm

This section presents the main result of the paper, that is, the UMA procedure, which is an answer to the problem stated above: UMA makes it possible to learn a multiclass linear predictor from $\mathcal{S}_y$ and the confusion information $\mathbf{C}$. In addition to the algorithm itself, this section provides theoretical results regarding the convergence and sample complexity of UMA.

As UMA is a generalization of the ultraconservative additive online algorithms introduced in section 2.3.2 to the case of noisy labels, we first and foremost recall the essential features of this family of algorithms. The rest of the section is then devoted to the presentation and analysis of UMA.

### 4.2.1 A Brief Reminder on Ultraconservative Additive Algorithms

As already stated, Ultraconservative additive online algorithm output multiclass linear predictors $h_{\mathbf{W}}$ as in Definition 2.3 and their purpose is therefore to compute a matrix $\mathbf{W} = [\mathbf{W}_{\cdot 1} \cdots \mathbf{W}_{\cdot Q}] \in \mathbb{R}^{D \times Q}$ of $Q$ weight vectors from some training sample $\mathcal{S} \doteq \{(\boldsymbol{x}_i, t_i)\}_{i=1}^{N}$. To do so, they implement the procedure depicted in Algorithm 2, which centrally revolves around the identification of an *error set* and its simple update: when processing a training pair $(\boldsymbol{x}, y(\boldsymbol{x}))$, they perform updates of the form

$$\mathbf{W}_{\cdot q} \leftarrow \mathbf{W}_{\cdot q} + \tau_q \boldsymbol{x} : q = 1, \ldots, Q$$

whenever the *error set* $\mathcal{E}$ (see Alg. 2) is not empty.

The main results regarding ultraconservative algorithms, which we will extend in our learning scenario is the one of Theorem 2.3 that is essentially a generalization of the well-known Block-Novikoff results [Block, 1962, Novikoff, 1962] (Theorem 1.3), which establishes a mistake bound for the Perceptron algorithm (an additive algorithm itself).

### 4.2.2 Main Result and High Level Justification

This section presents our main contribution, UMA, a theoretically grounded noise-tolerant multiclass algorithm depicted in Algorithm 4. UMA learns and outputs a matrix $\mathbf{W} = [\mathbf{W}_{\cdot 1} \cdots \mathbf{W}_{\cdot Q}] \in \mathbb{R}^{D \times Q}$ from a noisy training set $\mathcal{S}_y$ to produce the associated linear classifier

$$h_{\mathbf{W}}(\cdot) \doteq \arg\max_{q} \left\langle \mathbf{W}_{\cdot q}; \cdot \right\rangle \tag{4.3}$$

by iteratively updating the $\mathbf{W}_{\cdot q}$'s, whilst maintaining $\sum_q \mathbf{W}_{\cdot q} = \mathbf{0}$ throughout the learning process. As a new member of multiclass additive algorithms, we may readily recognize in step 8 through step 10 of Algorithm 4 the generic step sizes $\{\tau_q\}_{q \in \mathcal{Y}}$ promoted by ultraconservative algorithms (see Algorithm 2). An important feature of UMA is that it only uses information provided by $\mathcal{S}_y$ and does not make assumption on the accessibility to the noise-free dataset $\mathcal{S}_t$: the incurred pivotal difference with regular ultraconservative algorithms is that the update points used are now the computed (line 4 through line 7) $\boldsymbol{x}_{\text{up}}^{pq}$ vectors instead of the $\boldsymbol{x}_i$'s. Establishing that under some conditions UMA stops and provides a classifier with small risk when those update points are used is the purpose of the following subsections; we will also discuss the unspecified step 3, dealing with the selection step.

For the impatient reader, we may already leak some of the ingredients we use to prove the relevance of our procedure. Theorem 4.1, which shows the convergence of ultraconservative algorithms, rests on the analysis of the updates made when training examples are misclassified by the current classifier. The conveyed message is therefore that examples that are erred upon are central to the convergence analysis. It turns out that steps 4 through 7 of UMA (cf. Algorithm 4) construct a point $\boldsymbol{x}_{\text{up}}^{pq}$ that is, with high probability, mistaken on. More precisely, the true class $t(\boldsymbol{x}_{\text{up}}^{pq})$ of $\boldsymbol{x}_{\text{up}}^{pq}$ is $q$ and it is predicted to be of class $p$ by the current classifier; at the same time, these update vectors are guaranteed to realize a positive margin condition with respect to $\mathbf{W}^*$: $\left\langle \mathbf{W}_{\cdot q}^*; \boldsymbol{x}_{\text{up}}^{pq} \right\rangle > \left\langle \mathbf{W}_{\cdot k}^*; \boldsymbol{x}_{\text{up}}^{pq} \right\rangle$ for all $k \neq q$. The ultraconservative feature of the algorithm is carried by step 8 and step 10, which make it possible to update any prototype vector $\mathbf{W}_{\cdot r}$ with $r \neq q$ having an inner product $\left\langle \mathbf{W}_{\cdot r}; \boldsymbol{x}_{\text{up}}^{pq} \right\rangle$ with $\boldsymbol{x}_{\text{up}}^{pq}$ larger than $\left\langle \mathbf{W}_{\cdot q}; \boldsymbol{x}_{\text{up}}^{pq} \right\rangle$ (which should be the largest if a correct prediction was made). The reason why we have results 'with high probability' is because the $\boldsymbol{x}_{\text{up}}^{pq}$'s are sample-based estimates of update vectors known to be of class $q$ but predicted as being of class $p$, with $p \neq q$; computing the accuracy of the sample estimates is one of the important exercises of what follows. A control on the accuracy makes it possible for us to then establish the convergence of the proposed algorithm.

**Algorithm 4** UMA: Unconfused Ultraconservative Multiclass Algorithm.

**Require:** $\mathcal{S}_y = \{\boldsymbol{x}_i, y_i)\}_{i=1}^N$, confusion matrix $\mathbf{C} \in \mathbb{R}^{Q\times Q}$, and $\alpha > 0$
**Ensure:** $\mathbf{W} = [\mathbf{W}_{\cdot 1}, \ldots, \mathbf{W}_{\cdot Q}]$ and classifier $h_{\mathbf{W}}(\cdot) = \arg\max_q \langle \mathbf{W}_{\cdot q}; \boldsymbol{x} \rangle$

1: $\mathbf{W}_{\cdot k} \leftarrow 0, \forall k \in \mathbb{Y}$
2: **repeat**
3: select $p$ and $q$, $p \neq q$
4: compute set $\mathcal{A}_p^\alpha$ as

$$\mathcal{A}_p^\alpha \leftarrow \left\{\boldsymbol{x} | \boldsymbol{x} \in \mathcal{S}_y, \langle \mathbf{W}_{\cdot p}; \boldsymbol{x} \rangle - \langle \mathbf{W}_{\cdot k}; \boldsymbol{x} \rangle \geq \alpha : \forall k \neq p \right\}$$

5: for $k = 1, \ldots, Q$, compute $\theta_k^p$ as

$$\theta_k^p \leftarrow \frac{1}{N} \sum_{i=1}^N \mathbb{I}\left[y_i = k\right] \mathbb{I}\left[\boldsymbol{x}_i \in \mathcal{A}_p^\alpha\right] \boldsymbol{x}_i^\top : \forall k \in \mathbb{Y}$$

6: form $\Theta^p \in \mathbb{R}^{Q\times D}$ as

$$\Theta^p \leftarrow \left[\theta_1^p \cdots \theta_Q^p\right]^\top$$

7: compute the update vector $\boldsymbol{x}_{\text{up}}^{pq}$ according to ($[M]_q$ refers to the $q$th row of matrix $\mathbf{M}$)

$$\boldsymbol{x}_{\text{up}}^{pq} \leftarrow \left([\mathbf{C}^{-1}\Theta^p]_q\right)^\top$$

8: compute the error set $\mathcal{E}^\alpha(\boldsymbol{x}_{\text{up}}^{pq}, q)$ as

$$\mathcal{E}^\alpha(\boldsymbol{x}_{\text{up}}^{pq}, q) \leftarrow \{r \in \mathbb{Y} \backslash \{q\} : \langle \mathbf{W}_{\cdot r}; \boldsymbol{x}_{\text{up}}^{pq} \rangle - \langle \mathbf{W}_{\cdot q}; \boldsymbol{x}_{\text{up}}^{pq} \rangle \geq \alpha\}$$

9: **if** $\mathcal{E}^\alpha(\boldsymbol{x}_{\text{up}}^{pq}, q) \neq \emptyset$ **then**
10: compute some ultraconservative update steps $\tau_1, \ldots, \tau_Q$ such that:

$$\begin{cases} \tau_q = 1 \\ \tau_r \leq 0, \forall r \in \mathcal{E}^\alpha(\boldsymbol{x}_{\text{up}}^{pq}, q) \\ \tau_r = 0, \text{otherwise} \end{cases} \quad \text{and} \quad \sum_{r=1}^Q \tau_r = 0$$

11: perform the updates for $r = 1, \ldots, Q$:

$$\mathbf{W}_{\cdot r} \leftarrow \mathbf{W}_{\cdot r} + \tau_r \boldsymbol{x}_{\text{up}}^{pq}$$

12: **end if**
13: **until** $\|\boldsymbol{x}_{\text{up}}^{pq}\|_2$ is too small

### 4.2.3 With High Probability, $x_{\text{up}}^{pq}$ is a Mistake with Positive Margin

Here, we prove that the update vector $x_{\text{up}}^{pq}$ given in step 7 is, with high probability, a point on which the current classifier errs.

**Proposition 4.1** *Let $\mathbf{W} = [\mathbf{W}_{\cdot 1} \cdots \mathbf{W}_{\cdot Q}] \in \mathbb{R}^{D \times Q}$ and $\alpha > 0$ be fixed. Let $\mathcal{A}_p^\alpha$ be defined as in step 4 of Algorithm 4, i.e:*

$$\mathcal{A}_p^\alpha \doteq \left\{ \boldsymbol{x} | \boldsymbol{x} \in \mathcal{S}_y, \left\langle \mathbf{W}_{\cdot p}; \boldsymbol{x} \right\rangle - \left\langle \mathbf{W}_{\cdot k}; \boldsymbol{x} \right\rangle \geq \alpha : \forall k \neq p \right\}. \tag{4.4}$$

*that is, $\mathcal{A}_p^\alpha$ is the set of all examples predicted $p$ with margin at least $\alpha$.*

*For $k \in \mathbb{Y}$, $p \neq k$, consider the random variable $\theta_k^p$ ($\theta_k^p$ in step 5 of Algorithm 4 is a realization of this variable, hence the overloading of notation $\theta_k^p$):*

$$\theta_k^p \doteq \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[y_i = k\right] \mathbb{I}\left[\boldsymbol{x}_i \in \mathcal{A}_p^\alpha\right] \boldsymbol{x}_i^\top$$

*The following holds, for all $k \in \mathbb{Y}$:*

$$\mathbb{E}_{\mathcal{S}_y}\left[\theta_k^p\right] = \mathbb{E}_{\mathcal{S}_y \overset{i.i.d}{\sim} \mathscr{D}_y^N}\left[\theta_k^p\right] = \sum_{q=1}^{Q} \mathbf{C}_{kq} \mu_q^p, \tag{4.5}$$

*where*

$$\mu_q^p \doteq \mathbb{E}_{\mathscr{D}}\left[\mathbb{I}\left[t(X) = q\right] \mathbb{I}\left[X \in \mathcal{A}_p^\alpha\right] X^\top\right]. \tag{4.6}$$

*Proof.* Let us compute $\mathbb{E}_{\mathscr{D}_y}\left[\mathbb{I}\left[Y = k\right] \mathbb{I}\left[X \in \mathcal{A}_p^\alpha\right] X^\top\right]$:

$$\begin{aligned}
&\mathbb{E}_{\mathscr{D}_y}\left[\mathbb{I}\left[Y = k\right] \mathbb{I}\left[X \in \mathcal{A}_p^\alpha\right] X^\top\right] \\
&= \int_{\mathbb{X}} \sum_{q=1}^{Q} \mathbb{I}\left[q = k\right] \mathbb{I}\left[\boldsymbol{x} \in \mathcal{A}_p^\alpha\right] \boldsymbol{x}^\top \mathbb{P}_{\mathscr{D}_y}(Y = q | X = \boldsymbol{x}) d\mathscr{D}(\boldsymbol{x}) \\
&= \int_{\mathbb{X}} \mathbb{I}\left[\boldsymbol{x} \in \mathcal{A}_p^\alpha\right] \boldsymbol{x}^\top \mathbb{P}_{\mathscr{D}_y}(Y = k | X = \boldsymbol{x}) d\mathscr{D}(\boldsymbol{x}) \\
&= \int_{\mathbb{X}} \mathbb{I}\left[\boldsymbol{x} \in \mathcal{A}_p^\alpha\right] \boldsymbol{x}^\top \mathbf{C}_{kt(\boldsymbol{x})} d\mathscr{D}(\boldsymbol{x}) \qquad \text{(cf. eqn (4.2))} \\
&= \int_{\mathbb{X}} \sum_{q=1}^{Q} \mathbb{I}\left[t(\boldsymbol{x}) = q\right] \mathbb{I}\left[\boldsymbol{x} \in \mathcal{A}_p^\alpha\right] \boldsymbol{x}^\top \mathbf{C}_{kq} d\mathscr{D}(\boldsymbol{x}) \\
&= \sum_{q=1}^{Q} \mathbf{C}_{kq} \int_{\mathbb{X}} \mathbb{I}\left[t(\boldsymbol{x}) = q\right] \mathbb{I}\left[\boldsymbol{x} \in \mathcal{A}_p^\alpha\right] \boldsymbol{x}^\top d\mathscr{D}(\boldsymbol{x}) \\
&= \sum_{q=1}^{Q} \mathbf{C}_{kq} \mu_q^p,
\end{aligned}$$

where the last line comes from the fact that the classes are non-overlapping. The $N$ pairs $(X, Y)$ being identically and independently distributed gives the result. □

Intuitively, $\mu_q^p$ must be seen as an example of class $p$ which is erroneously predicted as being of class $q$. Such an example is precisely what we are looking for to update the current classifier; as expectations cannot be computed, the estimate $x_{\text{up}}^{pq}$ of $\mu_q^p$ is used instead of $\mu_q^p$.

**Proposition 4.2** *Let* $\mathbf{W} = [\mathbf{W}_{\cdot 1} \cdots \mathbf{W}_{\cdot Q}] \in \mathbb{R}^{D \times Q}$ *and* $\alpha \geq 0$ *be fixed. For* $p, q \in \mathbb{Y}$, $p \neq q$, $\boldsymbol{x}_{up}^{pq} \in \mathbb{R}^D$ *is such that*

$$\mathbb{E}_{\mathscr{D}_y}\left[\boldsymbol{x}_{up}^{pq}\right] = \mu_q^p \tag{4.7}$$

$$\left\langle \mathbf{W}_{\cdot q}^*; \mu_q^p \right\rangle - \left\langle \mathbf{W}_{\cdot k}^*; \mu_q^p \right\rangle \geq \gamma : \forall k \neq q, \tag{4.8}$$

$$\left\langle \mathbf{W}_{\cdot p}; \mu_q^p \right\rangle - \left\langle \mathbf{W}_{\cdot k}; \mu_q^p \right\rangle \geq \alpha : \forall k \neq p \tag{4.9}$$

*(Normally, we should consider the transpose of* $\mu_q^p$*, but since we deal with vectors of* $\mathbb{R}^D$ *—and not matrices— we abuse the notation and omit the transpose.)*

*This means that*

*i)* $t(\mu_q^p) = q$, i.e. *the 'true' class of* $\mu_q^p$ *is* $q$*;*

*ii) and* $h_{\mathbf{W}}(\mu_q^p) = p$*;* $\mu_q^p$ *is therefore misclassified by the current classifier* $h_{\mathbf{W}}$*.*

*Proof.* According to Proposition 4.1,

$$\mathbb{E}_{\mathscr{D}_y}\left[\Theta^p\right] = \mathbb{E}_{\mathscr{D}_y}\begin{bmatrix} \theta_1^p \\ \vdots \\ \theta_Q^p \end{bmatrix} = \begin{bmatrix} \mathbb{E}_{\mathscr{D}_y}\left[\theta_1^p\right] \\ \vdots \\ \mathbb{E}_{\mathscr{D}_y}\left[\theta_Q^p\right] \end{bmatrix} = \begin{bmatrix} \sum_{q=1}^Q \mathbf{C}_{1q}\mu_1^p \\ \vdots \\ \sum_{q=1}^Q \mathbf{C}_{Qq}\mu_Q^p \end{bmatrix} = \mathbf{C}\begin{bmatrix} \mu_1^p \\ \vdots \\ \mu_Q^p \end{bmatrix}$$

Hence, inverting $\mathbf{C}$ and extracting the $q$th of the resulting matrix equality gives that $\mathbb{E}\left[\boldsymbol{x}_{\text{up}}^{pq}\right] = \mu_q^p$.

Equation 4.8 is obtained thanks to Assumption 4.1 combined with equation (4.6) and the linearity of the expectation. Equation (4.9) is obtained thanks to equation (4.4) of $\mathcal{A}_p^\alpha$ (made of points that are predicted to be of class $p$) and the linearity of the expectation. □

The attentive reader may notice that Proposition 4.2 or, equivalently, step 7, is precisely the reason for requiring $\mathbf{C}$ to be invertible, as the computation of $\boldsymbol{x}_{\text{up}}^{pq}$ hinges on the resolution of a system of equations based on $\mathbf{C}$.

**Proposition 4.3** *Let* $\varepsilon > 0$ *and* $\delta \in (0; 1]$*. There exists a number*

$$N_0(\varepsilon, \delta, d, Q) = \mathcal{O}\left(\frac{1}{\varepsilon^2}\left[\ln\frac{1}{\delta} + \ln Q + d\ln\frac{1}{\varepsilon}\right]\right)$$

*such that if the number of training samples is greater than* $N_0$ *then, with high probability*

$$\left\langle \mathbf{W}_{\cdot q}^*; \boldsymbol{x}_{up}^{pq} \right\rangle - \left\langle \mathbf{W}_{\cdot k}^*; \boldsymbol{x}_{up}^{pq} \right\rangle \geq \gamma - \varepsilon : \forall k \neq \tag{4.10}$$

$$\left\langle \mathbf{W}_{\cdot p}; \boldsymbol{x}_{up}^{pq} \right\rangle - \left\langle \mathbf{W}_{\cdot k}; \boldsymbol{x}_{up}^{pq} \right\rangle \geq 0 : \forall k \neq p. \tag{4.11}$$

*Proof.* The existence of $N_0$ relies on pseudo-dimension arguments. We defer this part of the proof to Appendix A.2 and we will directly assume here that if $N \geq N_0$, then, with probability $1 - \delta$ for any $\mathbf{W}$, $\boldsymbol{x}_{\text{up}}^{pq}$.

$$\left| \left\langle \mathbf{W}_{\cdot p} - \mathbf{W}_{\cdot q}; \boldsymbol{x}_{\text{up}}^{pq} \right\rangle - \left\langle \mathbf{W}_{\cdot p} - \mathbf{W}_{\cdot q}; \mu_q^p \right\rangle \right| \leq \varepsilon \tag{4.12}$$

Proving equation (4.10) then proceeds by observing that

$$\left\langle \mathbf{W}^*_{\cdot q} - \mathbf{W}^*_{\cdot k}; \boldsymbol{x}^{pq}_{\text{up}} \right\rangle = \left\langle \mathbf{W}^*_{\cdot q} - \mathbf{W}^*_{\cdot k}; \mu^p_q \right\rangle + \left\langle \mathbf{W}^*_{\cdot q} - \mathbf{W}^*_{\cdot k}; \boldsymbol{x}^{pq}_{\text{up}} - \mu^p_q \right\rangle$$

bounding the first part using Proposition 4.1:

$$\left\langle \mathbf{W}^*_{\cdot q} - \mathbf{W}^*_{\cdot k}; \mu^p_q \right\rangle \geq \gamma$$

and the second one with equation (4.12). A similar reasoning allows us to get equation (4.11) by setting $\alpha \doteq \varepsilon$ in $\mathcal{A}^\alpha_p$. □

This last proposition essentially says that the update vectors $\boldsymbol{x}^{pq}_{\text{up}}$ that we compute are, with high probability, erred upon and realize a margin condition $\gamma - \varepsilon$.

Note that $\alpha$ is needed to cope with the imprecision incurred by the use of empirical estimates. Indeed, we can only approximate $\left\langle \mathbf{W}_{\cdot p}; \boldsymbol{x}^{pq}_{\text{up}} \right\rangle - \left\langle \mathbf{W}_{\cdot k}; \boldsymbol{x}^{pq}_{\text{up}} \right\rangle$ in equation (4.11) up to a precision of $\varepsilon$. Thus for the result to hold we need to have $\left\langle \mathbf{W}_{\cdot p}; \mu^p_q \right\rangle - \left\langle \mathbf{W}_{\cdot k}; \mu^p_q \right\rangle \geq \varepsilon$ which is obtained from equation (4.9) when $\alpha = \varepsilon$. In practice, this just says that the points used in the computation of $\boldsymbol{x}^{pq}_{\text{up}}$ are at a distance at least $\alpha$ from any decision boundaries.

**Remark 4.1** *It is important to understand that the parameter $\alpha$ helps us derive sample complexity results by allowing us to retrieve a linearly separable training dataset with* positive *margin from the noisy dataset. The theoretical results we prove hold for any such $\alpha > 0$ parameter and the smaller this parameter, the larger the sample complexity,* i.e., *the harder it is for the algorithm to take advantage of a training samples that meets the sample complexity requirements. In other words, the smaller $\alpha$, the less likely it is for* UMA *to succeed; yet, as shown in the experiments, where we use $\alpha = 0$,* UMA *continues to perform quite well.*

### 4.2.4 Convergence and Stopping Criterion

We arrive at our main result, which provides both convergence and a stopping criterion.

**Theorem 4.1** (COnvergence of UMA) *Under Assumptions 4.1, 4.2 and 4.3 there exists a number $N$, polynomial in $D, 1/\gamma, Q, 1/\delta$, such that if the training sample is of size at least $N$, then, with high probability $(1-\delta)$,* UMA *makes at most $\mathcal{O}(1/\gamma^2)$ updates.*

*Proof.* Let $\mathcal{S}_{\boldsymbol{x}_{\text{up}}}$ the set of all the update vectors $\boldsymbol{x}^{pq}_{\text{up}}$ generated during the execution of UMA and labeled with their *true* class $q$. Observe that, in this context, UMA (Alg. 4) behaves like a regular ultraconservative algorithm run on $\mathcal{S}_{\boldsymbol{x}_{\text{up}}}$. Namely: a) lines 4 through 7 compute a new point in $\mathcal{S}_{\boldsymbol{x}_{\text{up}}}$, and b) lines 8 through 10 perform an ultraconservative update step.

From Proposition 4.3, we know that with high probability, $w^*$ is a classifier with positive margin $\gamma - \varepsilon$ on $\mathcal{S}_{\boldsymbol{x}_{\text{up}}}$ and it comes from Theorem 2.3 that UMA does not make more than $\mathcal{O}(1/\gamma^2)$ mistakes on such dataset.

Because, by construction, we have that with high probability each element of $\mathcal{S}_{\boldsymbol{x}_{\text{up}}}$ is erred upon then $|\mathcal{S}_{\boldsymbol{x}_{\text{up}}}| \in \mathcal{O}(1/\gamma^2)$; that means that, with high probability, UMA does not make more than $\mathcal{O}(1/\gamma^2)$ updates.

All in all, after $\mathcal{O}(1/\gamma^2)$ updates, there is a high probability that we are not able to construct examples on which UMA makes a mistake or, equivalently, the conditional misclassification errors $\mathbb{P}(h_{\mathbf{W}}(X) = p|T = q)$ are all small. □

Even though UMA operates in a batch setting, it 'internally' simulates the execution of an online algorithm that encounters a new training point ($\boldsymbol{x}_{\text{up}}^{pq} \in \mathcal{S}_{\boldsymbol{x}_{\text{up}}}$) at each time step. To more precisely see how UMA can be seen as an online algorithm, it suffices to imagine it to be run in a way where each vector update is made after a chunk of $N$ (where $N$ is as in theorem 4.1) training data has been encountered and used to compute the next element of $\mathcal{S}_{\boldsymbol{x}_{\text{up}}}$. Repeating this process $\mathcal{O}(1/\gamma^2)$ times then guarantees convergence with high probability. Note that, in this scenario, UMA requires $N' = \mathcal{O}(N/\gamma^2)$ data to converge which might be far more than the sample complexity exhibited in theorem 4.1. Nonetheless, $N'$ still remains polynomial in $d$, $1/\gamma$, $Q$ and $1/\delta$. For more detail on this (online to batch conversion) approach, we refer the interested readers to [Blum et al., 1998].

### 4.2.5 Selecting $p$ and $q$

So far, the question of selecting good pairs of values $p$ and $q$ to perform updates has been left unanswered. Indeed, our results hold for *any* pair $(p, q)$ and convergence is guaranteed even when $p$ and $q$ are arbitrarily selected as long as $\boldsymbol{x}_{\text{up}}^{pq}$ is not $\mathbf{0}$. Nonetheless, it is reasonable to use heuristics for selecting $p$ and $q$ with the hope that it might improve the practical convergence speed.

On one hand, we may focus on the pairs $(p, q)$ for which the empirical misclassification rate

$$\hat{\mathbb{P}}_{X\sim\mathcal{S}_y}\{h_{\mathbf{W}}(X) \neq t(X)\} \doteq \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left[h_{\mathbf{W}}(\boldsymbol{x}_i) \neq t(\boldsymbol{x}_i)\right] \tag{4.13}$$

is the highest ($X \sim \mathcal{S}_y$ means that $X$ is randomly drawn from the uniform distribution of law $\boldsymbol{x} \mapsto N^{-1}\sum_{i=1}^{N}\mathbb{I}\left[\boldsymbol{x} = \boldsymbol{x}_i\right]$ defined with respect to training set $\mathcal{S}_y = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$). We want to favor those pairs $(p, q)$ because, i) the induced update may lead to a greater reduction of the error and ii) more importantly, because $\boldsymbol{x}_{\text{up}}^{pq}$ may be more reliable, as $\mathcal{A}_p^{\alpha}$ will be bigger.

On the other hand, recent advances in the passive aggressive literature [Ralaivola, 2012] have emphasized the importance of minimizing the empirical confusion rate, given for a pair $(p, q)$ by the quantity

$$\hat{\mathbb{P}}_{X\sim\mathcal{S}_y}\{h_{\mathbf{W}}(X) = p|t(X) = q\} \doteq \frac{1}{N_q}\sum_{i=1}^{N}\mathbb{I}\left[t(\boldsymbol{x}_i) = q, h_{\mathbf{W}}(\boldsymbol{x}_i) = p\right] \tag{4.14}$$

where

$$N_q \doteq \sum_{i=1}^{N}\mathbb{I}\left[t(\boldsymbol{x}_i) = q\right]$$

This approach is especially worthy when dealing with imbalanced classes and one might want to optimize the selection of $(p, q)$ with respect to the confusion rate.

Obviously, since the true labels in the training data cannot be accessed, neither of the quantities defined in equation (4.13) and equation (4.14) can be computed. Using a result provided in [Blum et al., 1998], which states that the norm of an update vector computed as $\boldsymbol{x}_{\text{up}}^{pq}$ directly provides an estimate of equation (4.13), we devise two possible strategies for selecting $(p, q)$:

$$(p, q)_{\text{error}} \doteq \arg\max_{(p,q)} \|\boldsymbol{x}_{\text{up}}^{pq}\|_2 \tag{4.15}$$

$$(p, q)_{\text{conf}} \doteq \arg\max_{(p,q)} \frac{\|\boldsymbol{x}_{\text{up}}^{pq}\|_2}{\hat{\boldsymbol{\pi}}_q} \tag{4.16}$$

where $\hat{\boldsymbol{\pi}}_q$ is the estimated proportion of examples of true class $q$ in the training sample. In a way similar to the computation of $\boldsymbol{x}_{\text{up}}^{pq}$ in Algorithm 4, $\hat{\boldsymbol{\pi}}_q$ may be estimated as follows:

$$\hat{\boldsymbol{\pi}}_q = \frac{1}{N}[\mathbf{C}^{-1}\boldsymbol{\Pi}^y]_q$$

where $\boldsymbol{\Pi}^y \in \mathbb{R}^Q$ is the vector containing the number of examples from $\mathcal{S}_y$ having noisy labels $1, \ldots, Q$, respectively.

The second selection criterion is intended to normalize the number of errors with respect to the proportions of different classes and aims at being robust to imbalanced data. Our goal here is to provide a way to take into account the class distribution for the selection of $(p, q)$. Note that this might be a first step toward transforming UMA into an algorithm for minimizing the confusion risk, even though additional (and significant) work is required to provably provide UMA with this feature.

On a final note, we remark that $(p, q)_{\text{conf}}$ requires additional precautions when used: when $(p, q)_{\text{error}}$ is implemented, $\boldsymbol{x}_{\text{up}}^{pq}$ is guaranteed to be the update vector of maximum norm among all possible update vectors, whereas this no longer holds true when $(p, q)_{\text{conf}}$ is used and if $\boldsymbol{x}_{\text{up}}^{pq}$ is close to $\mathbf{0}$ then there may exist another possibly more informative—from the standpoint of convergence speed—update vector $\boldsymbol{x}_{\text{up}}^{p'q'}$ for some $(p', q') \neq (p, q)$.

### 4.2.6 UMA and Kernels

Thus far, we have only considered the situation where linear classifiers are learned. There are however many learning problems that cannot be handled effectively without going beyond linear classification. A popular strategy to deal with such a situation is obviously to make use of kernels [Schölkopf and Smola, 2002] (see also section 2.1). In this direction, there are (at least) two paths that can be taken. The first one is to revisit UMA and provide a kernelized algorithm based on a dual representation of the weight vectors, as it is done with the kernel Perceptron (see [Cristianini and Shawe-Taylor, 2000] and section 2.1 example 2.3) or its close cousins (see, *e.g.* [Friess et al., 1998, Dekel et al., 2005, Freund and Schapire, 1999]). Doing so would entail the question of finding sparse expansions of the weight vectors with respect to the training data in order to contain the prediction

time and to derive generalization guarantees based on such sparsity: this is an interesting and ambitious research program on its own. A second strategy, which we make use of in the numerical simulations, is simply to build upon the idea of Kernel Projection Machines [Blanchard and Zwald, 2008, Takerkart and Ralaivola, 2011]: first, perform a Kernel Principal Component Analysis (shorthanded as kernel-PCA afterwards) with $D$ principal axes, second, project the data onto the principal $D$-dimensional subspace and, finally, run UMA on the obtained data. The availability of numerous methods to efficiently extract the principal subspaces (or approximation thereof) [Bach and Jordan, 2002, Drineas et al., 2006, Drineas and Mahoney, 2005, Stempfel and Ralaivola, 2007, Williams and Seeger, 2001] makes this path a viable strategy to render UMA usable for non-linearly separable concepts. This explains why we decided to use this strategy in the present work.

## 4.3 Experiments

In this section, we present results from numerical simulations of our approach and we discuss different practical aspects of UMA. The ultraconservative step sizes retained are those corresponding to a regular Perceptron: $\tau_p = -1$ and $\tau_q = +1$, the other values of $\tau_r$ being equal to 0.

Section 4.3.1 discusses robustness results, based on simulations conducted on synthetic data while Section 4.3.2 takes it a step further and evaluates our algorithm on real data, with a realistic noise process related to Example 4.1 (cf. Section 4.1).

We essentially use what we call the *confusion rate* as a performance measure, which is akin to the Frobenius norm of the (empirical) confusion risk defined in Section 3.2:

$$\frac{1}{\sqrt{Q}} \| R^{\mathbf{C}}_{\mathcal{S}_{\text{test}}} \|_{\text{F}}$$

Where $\mathcal{S}_{\text{test}}$ is an independent test set containing new, uncorrupted, data draw from $\mathscr{D}_t$. Note that similarly to the empirical risk, $R^{\mathbf{C}}_{\mathcal{S}_{\text{test}}}$ is simply the empirical penchant of the *confusion risk matrix* $R^{\mathbf{C}}_{\mathscr{D}_t}$ (see Section 3.2), and the $1/\sqrt{Q}$ factor ensure that the confusion rate is comprised within 0 and 1.

### 4.3.1 Toy dataset

We use a 10-class dataset with a total of roughly $1,000$ 2-dimensional examples uniformly distributed according to $\mathscr{U}$, which is the uniform distribution over the unit circle centered at the origin. Labelling is achieved according to equation (2.2) given a set of 10 weight vectors $\mathbf{W}_{\cdot 1}, \ldots, \mathbf{W}_{\cdot 10}$, which are also randomly generated according to $\mathscr{U}$; all these weight vectors have therefore norm 1. A margin $\gamma = 0.025$ is enforced in the generated data by removing examples that are too close to the decision boundaries—practically, with this value of $\gamma$, the case where three classes are so close to each other that no training example from one of the classes remained after enforcing the margin never occurred.

The learned classifiers are tested against a dataset of $10,000$ points that are distributed according to the training distribution. The results reported in the tables and graphics are averaged over 10 runs.

The noise is generated from the sole confusion matrix. This situation can be tough to handle and is rarely met with real data but we stick with it as it is a good example of a worst-case scenario.

**Experimental Protocols**

**Robustness to noise** We first (Fig. 4.1) evaluate the robustness to noise of UMA by running our algorithm with various confusion matrices. We uniformly draw a reference non-negative square matrix $\mathbf{M}$, the rows of $\mathbf{M}$ are then normalized, *i.e.* each entry of $\mathbf{M}$ is divided by the sum of the elements of its row, so $\mathbf{M}$ is a stochastic matrix. If $\mathbf{M}$ is not invertible it is rejected and we draw a new matrix until we have an invertible one. Then, we define $\mathbf{N}$ such that $\mathbf{N} = (\mathbf{M} - \mathbf{I})/10$, where $\mathbf{I}$ is the identity matrix of order $Q$; typically $\mathbf{N}$ has non-positive diagonal entries and non-negative off-diagonal coefficients. We will use $\mathbf{N}$ to parametrize a family of confusion matrices that have their most dominant coefficient to move from their diagonal to their off-diagonal parts. Namely, we run UMA 20 times with confusion matrices $\mathbf{C} \in \{\mathbf{C}_i \doteq \Omega(\mathbf{I} + i\mathbf{N})\}_{i=1}^{20}$, where $\Omega$ is a matrix operator which outputs a (row-)stochastic matrix: when applied on matrix $\mathbf{A}$, $\Omega$ replaces the negative elements of $\mathbf{A}$ by zeros and it normalizes the rows of the obtained matrix; note that $i = 10$ corresponds to the case where $\mathbf{C} = \mathbf{M}$. Equivalently, one can think of $\mathbf{C}_i$ as the weighted average between $\mathbf{I}$ and $\Omega(\mathbf{N})$ where $\mathbf{I}$ has a constant weight of 1 and $\Omega(\mathbf{N})$ is weighted by $i$. Note that, after some point, further increasing $i$ has little effect on $\mathbf{C}_i$ as it eventually converges to $\Omega(\mathbf{N})$. Figure 4.1 plots our results against the Frobenius norm of the diagonal-free confusion matrix $\mathbf{C}$, that is: $\|\mathbf{C} - \mathrm{diag}(\mathbf{C})\|_{\mathrm{F}}$ where $\mathrm{diag}(\mathbf{C})$ is defined as:

$$[\mathrm{diag}(\mathbf{C})]_{ij} \doteq \begin{cases} \mathbf{C}_{ij} & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases}$$

For the sake of comparison, we also have run UMA with a fixed confusion matrix $\mathbf{C} = \mathbf{I}$ on the same data. This amounts to running a Perceptron through the data multiple times and it allows us to have a baseline for measuring the improvement induced by the use of the confusion matrix.

**Robustness to the incorrect estimation of the confusion matrix.** The second experiment (Fig. 4.1) evaluates the robustness of UMA to the use of a confusion matrix that is not exactly the confusion matrix that describes the noise process corrupting the data; this will allow us to measure the extent to which a confusion matrix (inaccurately) estimated from the training data can be dealt with by UMA. Using the same notation as before, and the same idea of generating a random stochastic reference matrix $\mathbf{M}$, we proceed as follows: we use the given matrix $\mathbf{M}$ to corrupt the noise-free dataset and then, each confusion matrix from the family $\{\mathbf{C}_i\}_{i=1}^{20}$ is fed to UMA as if it were the confusion matrix governing the noise process. We introduce the notion of *approximation* factor $\rho$ as $\rho(i) \doteq 1 - i/10$, so that

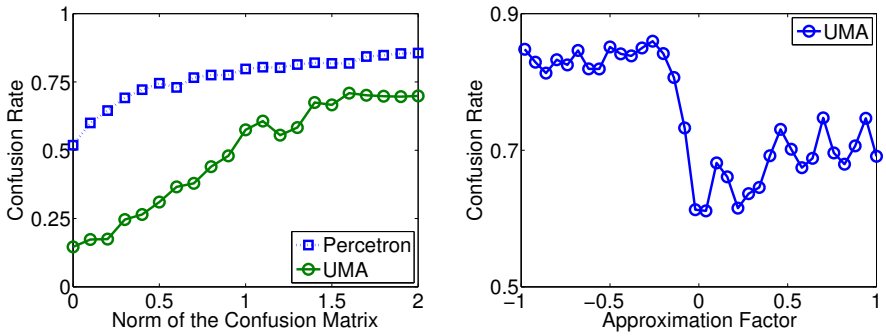



Figure 4.1 – *Left: Evolution of the confusion rate (y-axis) for different noise levels (x-axis).*
*Right: Evolution of the same quantity with respect to errors in the confusion matrix* $\mathbf{C}$ *(x-axis) measured by the approximation factor (see text).*

$\rho$ takes values in the set $\{-1, -0.9, \ldots, 0.9\}$. As reference, the limit case where $\rho = 1$—that is, $i = 0$—corresponds to the case where UMA is fed with the identity matrix $\mathbf{I}$, effectively being oblivious of any noise in the training set. More generally, the values of $\mathbf{C}$ are being shifted away from the diagonal as $\rho$ decreases, the equilibrium point being $\rho = 0$ where $\mathbf{C}$ is equal to the *true* confusion matrix $\mathbf{M}$. Consequently, a positive (resp. negative) approximation factor means that the noise is underestimated (resp. overestimated), in the sense that the noise process described by $\mathbf{C}$ would corrupt a lower (resp. higher) fraction of labels from each class than the *true* noise process applied on the training set, and corresponding to $\mathbf{M}$. Figure 4.1 plots the confusion rate against this approximation factor.

On Figure 4.1 we observe that UMA clearly provides improvement over the Perceptron algorithm for every noise level tested, as it achieves lower confusion rates. Nonetheless, its performance degrades as the noise level increases, going from a confusion rate of 0.5 for small noise levels—that is, when $\|\mathbf{C} - \mathrm{diag}(\mathbf{C})\|_{\mathrm{F}}$ is small—to roughly 2.25 when the noise is the strongest. Comparatively, the Perceptron algorithm follows the same trend, but with higher confusion rate, ranging from 1.7 to 2.75.

The second simulation (Fig. 4.1) points out that, in addition to being robust to the noise process itself, UMA is also robust to underestimated (approximation factor $\rho > 0$) noise levels, but not to overestimated (approximation factor $\rho < 0$) noise levels. Unsurprisingly, the best confusion rate corresponds to an approximation factor of 0, which means that UMA is using the true confusion matrix and can achieve a confusion rate as low as 1.8. There is a clear gap between positive and negative approximation factors, the former yielding confusion rates around 2.6 while the latter's are slightly lower, around 2.15. From these observations, it is clear that the approximation factor has a major influence on the performances of UMA.

### 4.3.2 Real data

#### Experimental Protocol

In addition to the results on synthetic data, we also perform simulations in a realistic learning scenario. In this section we are going to assume

that labelling examples is very expensive and we implement the strategy evoked in Example 4.1. More precisely, for a given dataset $\mathcal{S}_t$, proceed as follows:

1. Ask for a small number $M$ of examples for each of the $Q$ classes.
2. Learn a rough classifier[1] $y$ from these $Q \times M$ points.
3. Estimate the confusion $\mathbf{C}$ of $y$ on a small labelled subset $\mathcal{S}_{\text{conf}}$ of $\mathcal{S}$.
4. Predict the missing labels $y_i$ of $\mathcal{S}_y$ using $y$; thus, the $y$ acts as a corrupted concept .
5. Learn the final classifier $h_{\text{UMA}}$ from $\mathcal{S}$, $y$, $\mathbf{C}$ and measure its error rate.

One might wonder why we do not simply sample a very small portion of $\mathcal{S}_t$ in the first step. The reason is that in the case of very uneven classes proportions some of the classes may be missing in this first sampling. This is problematic when estimating $\mathbf{C}$ as it leads to a non-invertible confusion matrix. Moreover, the purpose of $y$ is only to provide a baseline for the labelling of $\mathcal{S}_y$, hence tweaking the class (im)balance in this step is not a problem.

In order to put our results into perspective, we compare them with results obtained from various algorithms. This allows us to give a precise idea of the benefits and limitations of UMA. Namely, we learn four additional classifiers: $h_{\mathcal{S}_y}$ is a regular Perceptron learned on $\mathcal{S}_y$ which is labelled with noisy labels from $y$, $h_{\mathcal{S}_{\text{conf}}}$ and $h_{\mathcal{S}_t}$ are trained with the correctly labelled training sets $\mathcal{S}_{\text{conf}}$ and $\mathcal{S}_t$ respectively and, lastly, $h_{\text{SSVM}}$ is a classifier produced by a multiclass semi-supervised *SVM* algorithm (SSVM, [Bennett and Demiriz, 1998]) run on $\mathcal{S}_{\text{SSVM}}$ which contains all the datapoints of $\mathcal{S}_t$ but only the labels of $\mathcal{S}_{\text{conf}}$ are provided. The performances achieved by $h_{\mathcal{S}_y}$ and $h_{\mathcal{S}_t}$ provide bounds for UMA's error rates: on the one hand, $h_{\mathcal{S}_y}$ corresponds to a worst-case situation, as we simply ignore the confusion matrix and use the regular Perceptron instead —arguably, UMA should perform better than this; on the other hand, $h_{\mathcal{S}_t}$ represents the best-case scenario for learning, when all the correct labels are available—the performance of $h_{\mathcal{S}_t}$ should always top that of UMA (and the performances of other classifiers). The last two classifiers, $h_{\mathcal{S}_{\text{conf}}}$ and $h_{\text{SSVM}}$, provide us with objective comparison measures. They are learned from the same data as UMA but use them differently: $h_{\mathcal{S}_{\text{conf}}}$ is learned from the reduced training set $\mathcal{S}_{\text{conf}}$ and $h_{\text{SSVM}}$ is output by a semi-supervised learning strategy that infers both $h_{\text{SSVM}}$ and the missing labels of $\mathcal{S}_{\text{SSVM}}$ and it totally ignores the predictions made by $y$. Note that according to the learning scenario we implement, we assume $\mathbf{C}$ to be estimated from raw data. This might not always be the case with real-world problems and $\mathbf{C}$ might be easier and/or less expensive to get than raw data; for instance, it might be deduced from expert knowledge on the studied domain. In that case, $h_{\mathcal{S}_{\text{conf}}}$ and $h_{\text{SSVM}}$ may suffer from not taking full advantage of the accurate information about the confusion.

[1] For the sake of self-containedness, we use UMA for this task (with $\mathbf{C}$ being the identity matrix). Remind that, when used this way, UMA acts as a regular Perceptron algorithm

**Datasets**

Our simulations are conducted on three different datasets. Each one with different features. For the sake of reproducibility, we used datasets that can be easily found on the *UCI Machine learning repository* [Bache and Lichman, 2013]. Moreover, these datasets correspond to tasks for which generating a complete, labelled, training set is typically costly because of the necessity of human supervision and subject to classification noise. The datasets used and their main features are as follows.

**Optical Recognition of Handwritten Digits.** This well-known dataset is composed of $8 \times 8$ grey-level images of handwritten digits, ranging from 0 to 9. The dataset is composed of $3,823$ images of 64 features for training, and $1,797$ for the test phase. We set $M$ to 10 for this dataset, which means that $y$ is learned from 100 examples only. $\mathcal{S}_{\text{conf}}$ is a sampling of 5% of $\mathcal{S}_t$. The classes are evenly distributed (see Figure 4.2). We handle the nonlinearity through the use of a Gaussian kernel-PCA(see section 4.2.6) to project the data onto a feature space of dimension 640.

**Letter Recognition.** The Letter Recognition dataset is another well-known pattern recognition dataset. The images of the letters are summarized into a vector of 16 attributes, which correspond to various primitives computed on the raw data. With $20,000$ examples, this dataset is much larger than the previous one. As for the Handwritten Digits dataset, the examples are evenly spread across the 26 classes (see Figure 4.2). We uniformly select $15,000$ examples for training and the remaining $5,000$ are used for test. We set $M$ to 50 as it seems that smaller values do not yield usable confusion matrices. We again sample 5% of the dataset to form $\mathcal{S}_{\text{conf}}$ and use, as before, a Gaussian kernel-based Kernel-PCA to (nonlinearly) expand the dimension of the data to $1,600$.

**Reuters.** The Reuters dataset is a nearly linearly-separable document categorization dataset of more than $300,000$ instances of nearly $47,000$ features each. For size reasons we restrict ourselves to roughly $15,000$ examples for training, and $15,000$ other for test. It occurs that some classes are so underrepresented that they are flooded by the noise process and/or do not appear in $\mathcal{S}_{\text{conf}}$, which may lead to a non-invertible confusion matrix. We therefore restrict the dataset to the 9 largest classes. One might wonder whether doing so erases class imbalance. This is not the case as, even this way, the least represented class accounts for roughly 500 examples while this number reaches nearly $4,000$ for the most represented one (see Figure 4.2). Actually, these 9 classes represent more than 70 percent of the dataset, reducing the training and test sets to approximately $11,000$ examples each. We do not use any kernel for this dataset, the data being already near to linearly-separable. Also, we sample $\mathcal{S}_{\text{conf}}$ on 5% of the training set and we set $M = 20$.

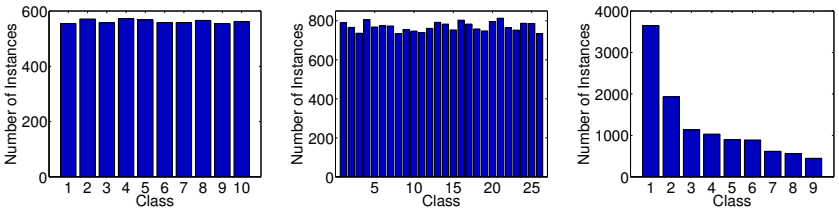



Figure 4.2 – *Class distribution for the three datasets.*

**Results**

Figure 4.3 presents the misclassification error rates averaged on 10 runs. Keep in mind that we have not conducted a very thorough optimization of the hyper-parameters as the point here is essentially to compare UMA with the other algorithms. Additionally, we also report the error rates of $h_{\text{SSVM}}$ when trained on the kernelized data with all dimensions, that is the kernelized data before we project them onto their $D$ principal components. Because the projection step is indeed unnecessary with SSVM, this will give us insights on the error due to the Kernel-PCA step. Comparing the first and the last columns of figure 4.3, it appears that UMA always induces a slight performance gain, *i.e.* a decrease of the misclassification rate, with respect to $h_{\mathcal{S}_y}$.

From the second and third columns of figure 4.3, it is clear that the reduced number of examples available to $h_{\mathcal{S}_{\text{conf}}}$ induces a drastic increase in the misclassification rate with respect to $h_{\mathcal{S}_t}$ which is allowed to use the totality of the dataset during the training phase.

Comparing UMA and $h_{\mathcal{S}_{\text{conf}}}$ in figure 4.3 (fifth and second columns), we observe that UMAachieves lower misclassification rates on the Handwritten Digits and Letter Recognition datasets but a higher misclassification rate on Reuters. This is likely related to the strong class imbalance in the dataset. Indeed, some classes are overly represented, accounting for the vast majority of the whole dataset (see Fig. 4.2). Because $\mathcal{S}_{\text{conf}}$ is uniformly sampled from the main dataset, $h_{\mathcal{S}_{\text{conf}}}$ is trained with a lot of examples from the overrepresented classes and therefore it is very effective, in the sense that it achieves a low misclassification rate, for these overrepresented classes; this, in turn, induces a (global) low misclassification rate, as possibly high misclassification rates on underrepresented classes are countervailed by theirs accounting for a small portion of the data. On the other hand, because of this disparity in class representation, the slightest error in the confusion matrix, granted it involves one of these overrepresented classes, may lead to a significant increase of the misclassification rate. In this regard, UMA is strongly disadvantaged with respect to $h_{\mathcal{S}_{\text{conf}}}$ on the Reuters dataset and it is the cause of the reported results.

The error rates for the SSVM and UMA classifiers are close for the Reuters and Handwritten Digits datasets whereas UMA has a clear advantage on the Letter Recognition problem. On the other hand, note that we used the SSVM method in conjunction with a Kernel-PCA for the sake of comparison with UMA in its kernelized form. The last column of figure 4.3 tends to confirm that this projection strategy increases the error rate of $h_{\text{SSVM}}$. Also, reminds that the value of $M$ does not impact the performances of $h_{\text{SSVM}}$ but has a significant effect on UMA, even though

| Dataset | $h_{\mathcal{S}_y}$ | $h_{\mathcal{S}_{\text{conf}}}$ | $h_{\mathcal{S}_t}$ | $h_{\text{SSVM}}$ | UMA | $h_{\texttt{SSVM}}$ (no K-PCA) |
|---|---|---|---|---|---|---|
| Handwritten Digits | 0.25 | 0.21 | 0.04 | 0.15 | 0.16 | 0.07 |
| Letter Recognition | 0.35 | 0.36 | 0.23 | 0.49 | 0.33 | 0.18 |
| Reuters | 0.30 | 0.17 | 0.01 | 0.22 | 0.21 | 0.22 |

Figure 4.3 – *Misclassification rates of different algorithms.*

UMA never uses these labelled data. For instance, on the Reuters datasets, increasing $M$ from 20 to 70 reduces UMA's error rate by nearly 0.1 (see the error rates of figure 4.4 ($m = 70$) when the size of labelled data is close to 550, that is 5% of the whole dataset). Despite our efforts to keep $M$ as small as possible, we could not go under $M = 50$ for the Letter Recognition dataset without compromising the invertibility of the confusion matrix. The simple fact that an unusually high number of examples are required to simply learn a rough classifier asserts the complexity of this dataset. Moreover, the fact that $h_{\mathcal{S}_y}$ also outperforms $h_{\text{SSVM}}$ implies that the labels fed to UMA are already mostly correct, and, according to our working assumptions, this is the most favorable setting for UMA.

Nonetheless, the disparities between UMA and $h_{\mathcal{S}_{\text{conf}}}$ deserve more attention. Indeed, the same data are being used by both algorithms, and one could expect more closeness in the results. To get a better insight on what is occurring, we have reported the evolution of the error rate of these two algorithms with respect to the sampling size of $\mathcal{S}_{\text{conf}}$ in Figure 4.4. We can see that UMA is unaffected by the size of the sample, essentially ignoring the possible errors in the confusion matrix on small samples. This reinforces our previous results showing that UMA is robust to errors in the confusion matrix. On the other hand, with the addition of more samples, the refinement of the confusion matrix does not allow UMA to compete with the value of additional (correctly) labelled data and eventually, when the size of $\mathcal{S}_{\text{conf}}$ grows, $h_{\mathcal{S}_{\text{conf}}}$ performs better than UMA. This points toward the idea that the aggregated nature of the confusion matrix incurs some loss of relevant information for the classification task at hand, and that a more accurate estimate of the confusion matrix, as induced by, *e.g.*, the use of larger $\mathcal{S}_{\text{conf}}$, may not compensate for the information provided by additional raw data.

Building on this observation, we go a step further and replicate this experiment for all of the three datasets; only this time we track the performances of $h_{\text{SSVM}}$ instead. The results are plotted on Figure 4.5. For the three datasets, we observe the same behavior as before. Namely, UMA is able to maintain a low error rate even with a very small size of $\mathcal{S}_{\text{conf}}$. On the other hand, UMA does not benefit as much as other methods from a large pool of labelled examples. In this case, UMA quickly stabilizes while, to the contrary, the SSVM method starts at a fairly high error rate and keeps improving as more labelled examples are available.

Beyond this, it is important to recall that UMA never uses the labels of $\mathcal{S}_{\text{conf}}$ (those are only used to estimate the confusion matrix, not the classifier (refer to Section 4.3.2 for the detailed learning protocol). While refining the estimation of $\mathbf{C}$ is undoubtedly useful, a direction toward substantial performance gains should revolve around the combination of both this refined estimation of $\mathbf{C}$ *and* the use of the correctly labelled training

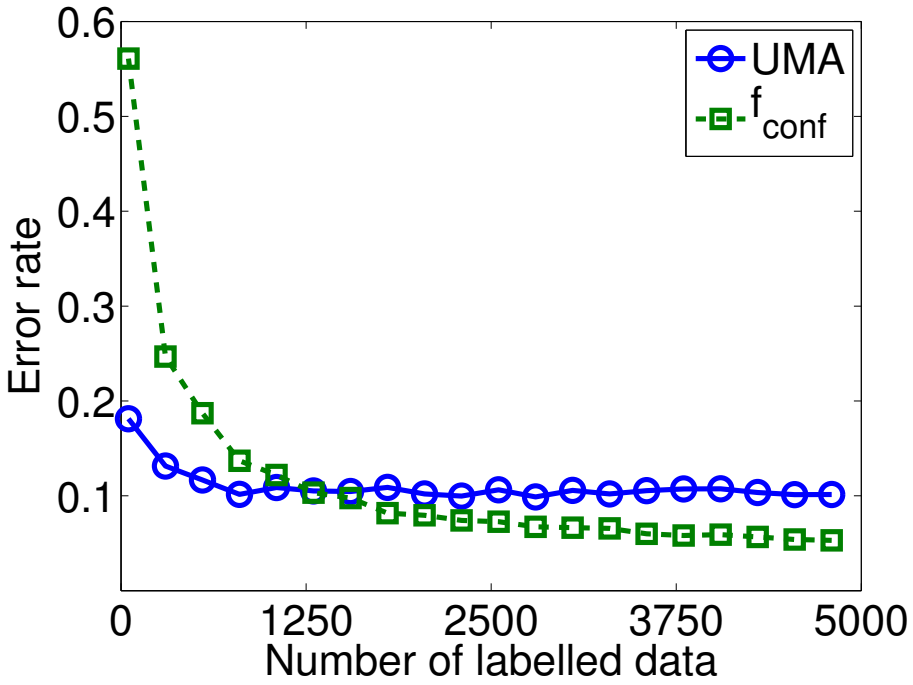



Figure 4.4 – *Error rate of* UMA *and* $h_{\mathcal{S}_{conf}}$ *with respect to the sampling size. Reuters dataset with* $m = 70$ *for the sake of figure's readability.*

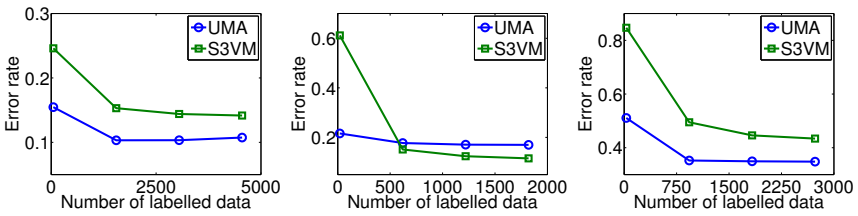



Figure 4.5 – *Error rates for the Reuters (left), optical digit recognition (center) and letter (right) datasets with respect to the size of* $\mathcal{S}_{conf}$*. Average over* 15 *runs.*

set $\mathcal{S}_{\text{conf}}$. This is a research subject on its own that we leave for future work.

All in all, the reported results advise us to prefer UMA over other available methods when the amount of labelled data is particularly small, in addition, obviously, to the motivating case of the present work where the training data are corrupted and the confusion matrix is known. Also, another interesting finding we get is that even a rough estimation of the confusion matrix is sufficient for UMA to behave well.

Finally, we investigate the impact of the selection strategy of $(p, q)$ on the convergence speed of UMA (see Section 4.2.5). We use three variations of UMA with different strategies for selecting $(p, q)$ (error, confusion, and random) and monitor each one along the learning process on the Reuters dataset. The error and confusion strategies are described in Section 4.2.5 and the random strategy simply selects $p$ and $q$ at random.

From Figure 4.6, which reports the misclassification rate and the confusion rate along the iterations, we observe that both performance measures evolve similarly, attaining a stable state around the 30th iteration. The best strategy depends on the performance measure used, even though re-

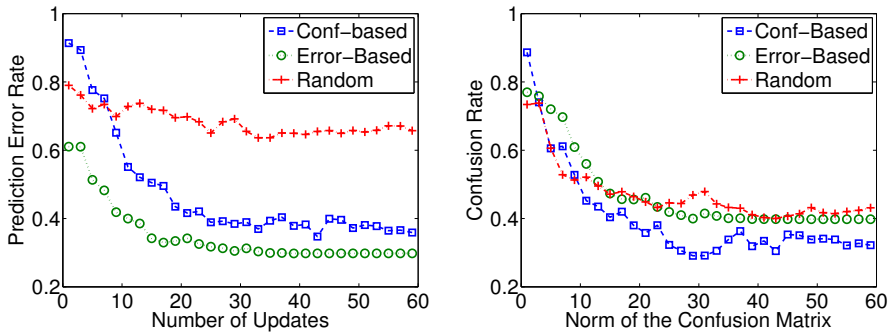



Figure 4.6 – *Error and confusion rates on Reuters dataset with various update strategies.*

gardless of the performance measure used, we observe that the random selection strategy leads to a predictor that does not achieve the best performance measure (there is always a curve beneath that of the random selection procedure), which shows that it not an optimal selection strategy.

As one might expect, the confusion-based strategy performs better than the error-based strategy when the confusion rate is retained as a performance measure, while the converse holds when using the error rate. This observation motivates us to thoroughly study the confusion-based strategy in a near future as being able to propose methods robust to class imbalance is a particularly interesting challenge of multiclass classification.

The plateau reached around the 30th iteration may be puzzling, since the studied dataset presents no positive margin and convergence is therefore not guaranteed. One possible explanation for this is to see the Reuters dataset as linearly separable problem corrupted by the effect of a noise process, which we call the *intrinsic noise process* that has structural features 'compatible' with the classification noise. By this, we mean that there must be features of the intrinsic noise such that, when additional classification noise is added, the resulting noise that characterizes the data is similar to a classification noise, or at least, to a noise that can be naturally handled by UMA. Finding out the family of noise processes that can be combined with the classification noise—or, more generally, the family of noise processes themselves—without hindering the effectiveness of UMA is one research direction that we aim to explore in a near future.

## 4.4 General Conclusion

### 4.4.1 Discussion and Afterthoughts

In this section we shall discuss some more of UMA characteristics and noticeable aspects. While this is not bound to the main exposition of UMA, we think some remarks are useful addition that bring contextualization and help in establishing what is the potentiality of our work in the fields of learning noisy classifier and multiclass classification.

One of the core ideas behind UMA, namely, the computation of the update vector $\boldsymbol{x}_{\text{up}}^{pq}$, is not tied to the additive update scheme. Thus, as long as the assumption of linear separability holds, the very same idea can be

used to render a wide variety of algorithms robust to noise by iteratively generating a noise-free training set with the consecutive values of $\boldsymbol{x}_{\text{up}}^{pq}$. Although, every computation of a new $\boldsymbol{x}_{\text{up}}^{pq}$ requires learning a new classifier to start with. This may eventually incur prohibitive computational costs when applied to batch methods (as opposed to online methods) which are designed to process the entirety of the dataset at once[2].

UMA takes advantage of the online scheme of additive algorithms and avoids this problem completely. Moreover, additive algorithms are designed to directly handle multiclass problem rather than having recourse to a bi-class mapping. The end-results of this are tightened theoretical guarantees and a convergence rate that does not depend of $Q$, the number of classes. Besides, UMA can be directly used with any additive algorithms, allowing to handle noise with multiple methods without further computational burden.

While we provide sample complexity analysis, it should be noted that a tighter bound can be derived with specific multiclass tools, such as the Natarajan's dimension (see [Daniely et al., 2011] for example), which allow to better specify the expressiveness of a multiclass classifier. However, this is not the main focus of this work and our results are based on simpler tools.

A complement to this work we want to investigate at a later time is a way to properly tackle near-linear problems (such as Reuters). As for now the algorithm already does a very good jobs due to its noise robustness. However more work has to be done to derive a proper way to handle cases where a perfect classifier does not exist. We think there are great avenues for interesting research in this domain with an algorithm like UMA and we are curious to see how this present work may carry over to more general problems.

### 4.4.2 Conclusion

In this chapter, we have proposed a new algorithm, UMA —for Unconfused Multiclass Additive algorithm— to cope with noisy training examples in multiclass linear problems. As its name indicates, it is a learning procedure that extends the (ultraconservative) additive multiclass algorithms introduced in section 2.3.2; to handle noisy datasets, it only requires the information about the confusion matrix that characterizes the mislabelling process. This is, to the best of our knowledge, the first work where the confusion matrix is used as a way to handle noisy label in multiclass problems.

As such, UMA provide a first positive answer in our capability to cope with unreliable or missing data efficiently and thus provides closure to part II. In part III will take an altogether different angle on the more general problem of label unavailability and explore how to adapt the theory and the setting of learning when obtaining a full training set of labelled samples is prohibitively costly.

[2]Nonetheless, from a purely theoretical point of view, UMA makes at most $O(1/\gamma^2)$ mistakes (see theorem 4.1) and computing $\boldsymbol{x}_{\text{up}}pq$ can be done in $O(N)$ time. Therefore, polynomial batch methods do not suffer much from this as their overall execution time is still polynomial.

# Part III

# Active Learning and Version Space

# From Linear Classification to Localization Problems

5

Contents



In this chapter we shall take a step aside and forget about Machine Learning for a moment. Indeed, we will essentially discuss the matter of localization problems and how one can solve them through Cutting Planes algorithms. The relevance of this chapter with respect to the last third of this thesis, that is part III, comes from the similarities between localization and learning problems. More precisely, the premise of this chapter is that machine learning and localization frameworks are two different perspectives on the same problem. Notably, the three last chapters of this thesis share the idea that we can leverage the Cutting Planes theory developed in computational geometry to devise theoretically sound and efficient Machine Learning methods. Given this general synopsis for part III this chapter aims at giving a gentle introduction to the formalism of Cutting Planes and localization problems as well as establishing the theoretical background needed for the two next chapters to unfold seamlessly. Additionally, theorem 5.2 [Louche and Ralaivola, 2015a] is a new addition to the theory of centers of gravity and one of the contributions of this thesis. We decided to expound it in this chapter because of its relevance to

computational geometry and centers of gravity methods even though we will not need it until chapter 7.

The first section will discuss in detail how the two settings we will consider can be thought as equivalent, and we will make explicit the link between machine learning and localization problems. The second section introduces Cutting Planes methods and discusses some of the properties of these methods that the reader should be aware of. Finally, the last section focuses on a key geometrical notion in Cutting Planes theory, that is the idea of centroid and more precisely, the notion of center of gravity.

## 5.1 An introduction to Localization Problem

As announced in the introduction, this chapter deals mostly with computational geometry, and more precisely localization problems. The goal of this section is to motivate and discuss the relevance of localization problems relatively to the problem of bi-class classification.

### 5.1.1 Motivating the Setting

#### The problem so far

Let us first recap the learning problem we want to tackle. Contrary to part II we will take a step back to binary classification as it was defined in chapter 1. More formally, the setting we consider consists in a $D$-dimensional input space $\mathbb{X}$ and an output space $\mathbb{Y}$ where $\mathbb{X}$ is a vector space with inner product, generally akin to $\mathbb{R}^D$ and $\mathbb{Y} \doteq \{+1; -1\}$. The goal of learning as it has been defined previously is to learn some concept function $t$ that maps $\mathbb{X}$ to $\mathbb{Y}$ based on the information contained in some training set $\mathcal{S} \doteq \{(\boldsymbol{x}_i, t_i)\}_{i=1}^N$ where the pairs $(\boldsymbol{x}_i, t_i)$ are independently drawn from a joint distribution $\mathscr{D}_t$. More precisely, the crux of this setting is to find some point $\boldsymbol{w} \in \mathbb{X}$ such that the hyperplane $h_{\boldsymbol{w}}$ of normal vector $\boldsymbol{w}$ separates the the points mapped to $(-1)$ from the points mapped to $(+1)$. In other words, we define the classifier $h_{\boldsymbol{w}}(\cdot)$ as

$$h_{\boldsymbol{w}}(\cdot) \doteq \text{sign}(\langle \boldsymbol{w}; \cdot \rangle)$$

and ask for $h_{\boldsymbol{w}}$ to mimic the concept $t$. Under P.A.C.-learnability assumption, it is known (see the discussion in section 1.2.3) that we can assert the quality of $h_{\boldsymbol{w}}$ over $\mathscr{D}_t$ from its error rate on $\mathcal{S}$ given that $\mathcal{S}$ is large enough. However, allowing for more expressive classifier via kernelization (Section 2.1) requires finer optimality condition such as Compression (Section 2.2) or Large Margin (Section 1.3) properties. Finally, we keep our setting noise free (Section 3.1) and will assume that a suitable $h$ that perfectly mimics $t$ on $\mathcal{S}$ can always be found, in other words there always exists a vector $\boldsymbol{w}$ such that $R_{\mathcal{S}}^{l_{0-1}}(h_{\boldsymbol{w}}) = 0$.

Therefore, the problem we really want to tackle in bi-class classification so far is the one of finding a vector $\boldsymbol{w} \in \mathbb{X}$ such that, for any point $\boldsymbol{x} \in \mathcal{S}$, $\boldsymbol{w}$ realizes a positive dot product with $t(\boldsymbol{x})\boldsymbol{x}$: $t(\boldsymbol{x}) \langle \boldsymbol{w}; \boldsymbol{x} \rangle \geq 0$. An alternate

take on this problem is to define a matrix $\mathbf{X} \in \mathbb{X}^N$ such that

$$\mathbf{X} \doteq \begin{bmatrix} t_1 \boldsymbol{x}_1^\top \\ \vdots \\ t_N \boldsymbol{x}_N^\top \end{bmatrix}$$

and to rewrite the problem of learning as:

$$\text{find} \quad w \quad \text{s.t.} \begin{cases} \boldsymbol{w} \in \mathbb{X} \\ \mathbf{X}\boldsymbol{w} \geq \mathbf{0} \end{cases} \tag{5.1}$$

Note that, in the above formulation, the class labels are integrated within the matrix $\mathbf{X}$. As such, $\mathbf{X}$ can be thought as the matrix containing the data of the problem, this formalism is notably related to the *normal form* of a training set previously introduced in definition 3.1. For reminder, the normal form of $\mathcal{S}$ is a dataset $\mathcal{S}'$ where all points have a positive label which is defined as : $\mathcal{S}' \doteq \{(\boldsymbol{x}'_i, +1) : \boldsymbol{x}'_i = t_i \boldsymbol{x}_i\}_{i=1}^N$

**An alternative interpretation of learning**

Problem (5.1) belongs to the class of linear programming problems (more precisely, feasibility problems), a widely studied and fundamental class of problem of computer science. It is known that learning problems can be seen as peculiar form of linear problems [Dunagan and Vempala, 2008] and linear programming methods are known to be efficient in machine learning [Bennett and Parrado-Hernández, 2006] although they usually lack the more advanced design feature of modern machine learning algorithm. As such, optimization methods for linear programming (or quadratic programming in the SVM case) are often relegated as algorithmic tools and rarely studied for their inherent learning properties.

So far, the problem of learning we have been interested in has always been thought as learning a separating hyperplane. The underlying semantic being that data are points in $\mathbb{X}$ labeled either $+1$ or $-1$, thus the intuitive goal behind binary classification is to differentiate between points labeled $+1$ and $-1$ or, in other words, to find a hyperplane such that points with different labels lie on different sides of the hyperplane. As discussed in section 3.1, the introduction of the normal form of $\mathcal{S}$, noted $\mathcal{S}'$, changed a little this interpretation by introducing dataset with only positive labels; notably as far as learning is concerned, $\mathcal{S}$ and $\mathcal{S}'$ are functionally the same.

The key motivation for this chapter is that, except from our inherent interpretation of what a learning task is, there is no actual differences between a classifier and a datapoint. Indeed, since section 1.1.2 and definition 1.1 we have build our settings around the fact that $\mathbb{H} \equiv \mathbb{X}$ through the used of the inner product as the pivotal element of the decision rule and although it seems natural to think of data as points and classifiers as hyperplanes, the semantics of the two can be switched freely. Namely, let us consider a *dual* interpretation of learning, that is where the matrix $\mathbf{X}$ is thought as a collection of hyperplanes, defined by their normal vectors, and $\boldsymbol{w}$ is akin to a point in $\mathbb{X}$. The problem of learning thus become the problem of finding a point $\boldsymbol{w}$ such that $\forall i \in [N] \, \langle \boldsymbol{w}; \mathbf{X}_{i\cdot} \rangle \geq 0$. In other

words, we want $\boldsymbol{w}$ to lie on the positive side of every hyperplanes defined in $\mathbf{X}$ at once (see also figure 5.1). As a matter of fact, this corresponds to a well-studied geometrical problem known as localization.

**Version space**

The crux of localization problems is to localize a point $\boldsymbol{w}$ within a convex *version space* $\mathcal{W}_0$. More formally, $\mathcal{W}_0$ is defined from a collection of hyperplanes $h_{\boldsymbol{x}1}, \dots, h_{\boldsymbol{x}_N}$ as the intersection of the positive halfspaces induced by $\mathbf{X}$, that is

$$\mathcal{W}_0(\mathbf{X}) \doteq \{\boldsymbol{w} \in \mathbb{X} : \forall i \in [N] \, \langle \boldsymbol{w}; \mathbf{X}_{i\cdot} \rangle \geq 0\}$$

and we can rewrite the localization problem as

$$\text{find} \quad w \quad \text{s.t.} \quad \boldsymbol{w} \in \mathcal{W}_0 \tag{5.2}$$

where we dropped the dependency over $\mathbf{X}$ for $\mathcal{W}_0$.

Thus, solving the localization problem (5.2) amounts to solving the linear problem (5.1) which, in turn, is equivalent to our learning problem.

Nonetheless, before going further down this road, we shall discuss a little more the specifics of problem (5.2). First, we may note that in our case $\mathcal{W}_0$ is unbounded, this is because all hyperplanes are defined from a normal vector only and as such, they all go through the origin of $\mathbb{X}$. Ultimately, this $\mathcal{W}_0$ being unbounded has to do with scale insensitiveness and for any point $\boldsymbol{w} \in \mathcal{W}_0$ note that for any $\lambda > 0$ and $i \in [N]$, $\langle \boldsymbol{w}; \mathbf{X}_{i\cdot} \rangle \Leftrightarrow \langle \lambda \boldsymbol{w}; \mathbf{X}_{i\cdot} \rangle$. Said otherwise, $\mathcal{W}_0$ is unbounded because, by construction of the learning problem, any rescaling of a valid classifier $\boldsymbol{w}$ is also valid. While this is a desirable property in the case of learning, dealing with unbounded version space will prove difficult. We refer to the discussion found in [Herbrich et al., 2001] in this matter and argue that if we enforce a Euclidean norm of 1 on $\mathcal{W}_0$, that is if we consider the set $\mathcal{W}_0 \cap \{\boldsymbol{w} : \|\boldsymbol{w}\|_2^2 = 1\}$ we have a convex and compact set over the geodesic that is the $D$-dimensional sphere of unit Euclidean norm. Although transposing our reasoning onto geodesic geometry is feasible and lead to a definition of the version space that is compact, with respect to said geodesic, this is an arguably impractical solution as underlined by [Herbrich et al., 2001]. Instead, we follow the solution proposed in [Minka, 2001a, Minka, 2001b, Herbrich et al., 2001] and adopt a more practical solution by enforcing a maximal norm over $\mathcal{W}_0$ thus defining a *restricted* version space $\mathcal{W}$ (figure 5.2)

$$\mathcal{W} \doteq \mathcal{W}_0 \cap \mathcal{B}_1$$

where $\mathcal{B}_1$ denotes the ball of unit Euclidean norm $\mathcal{B}_1 \doteq \{\boldsymbol{w} : \|\boldsymbol{w}\|_2^2 \leq 1\}$. What we lose here is that $\mathcal{W}$ is no longer defined in terms of hyperplanes only, although this can be easily

accounted for; moreover there is a strong connection between $\mathcal{W}$ as it is defined here and the geodesic approach, thus for most of what is to come reasoning over $\mathcal{W}$ and projecting over the geodesic afterwards (through normalization) is a sound solution to circumvent computationally costly geodesic calculus [Minka, 2001b, Herbrich et al., 2001].

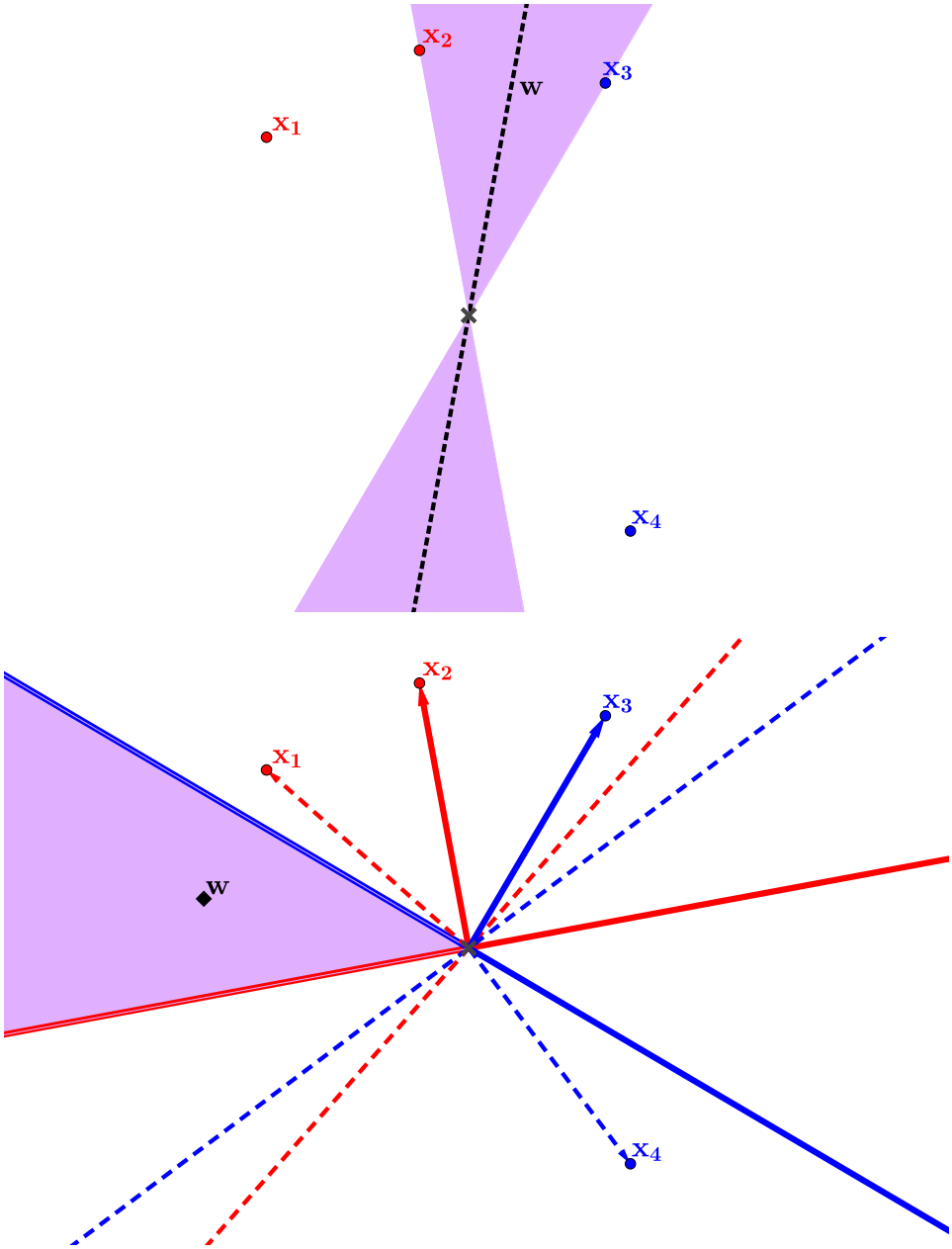



Figure 5.1 – *An illustration of the equivalence between a learning problem (top) and a localization problem (below) with two positive data (red) and two negative ones (blue). On the top picture, the purple area represent the set of separating hyperplane and $\boldsymbol{w}$ is one possible solution. Remark that on this example $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ are enough to fully describe the solution set. The bottom picture represents the equivalent localization problem, where each point is taken as a hyperplane normal vector —with negative points being first reversed. Again, the purple area in which $\boldsymbol{w}$ lives is only defined by $\boldsymbol{x}_2$ and $x_3$'s hyperplanes and the point $\boldsymbol{w}$ is an example of valid classifier.*

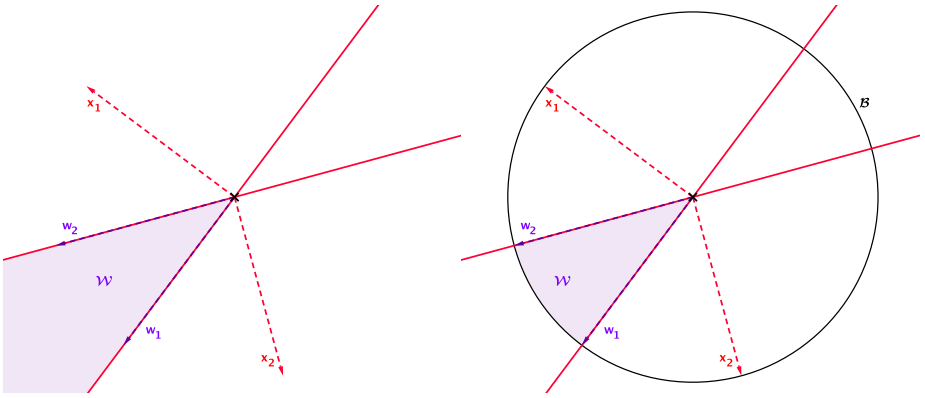



Figure 5.2 – *An illustration of a two dimensional version space with and without the restriction to the unit ball.*

Therefore, the localization problem we shall consider is the one of localizing a point $\boldsymbol{w}$ within some compact convex set $\mathcal{W}$, where $\mathcal{W}$ is defined by a collection of hyperplane whose normal vectors are in $\mathbf{X}$ and a quadratic constraint over the norm of $\boldsymbol{w}$.

$$\text{find} \quad w \quad \text{s.t.} \quad \boldsymbol{w} \in \mathcal{W} \tag{5.3}$$

In addition, note that solving the localization problem (5.3) yields a solution to the learning problem define in section 5.1.1. Also note that the reverse may be not true, yet any solution of the learning problem can be trivially rescaled to yield a solution to the localization problem. Hence, in practice, one can go from one problem to the other seamlessly.

## 5.2 Cutting Planes Algorithms

### 5.2.1 A general introduction to Cutting Planes Algorithms

*Cutting Planes* algorithms are a family of optimization methods that have reliably been used in machine learning as solver methods [Franc and Sonnenburg, 2009, Teo et al., 2010] for machine learning problems such as SVMs and regularized risk functionals. Namely, they are based on a geometric approach to linear problems, and try to iteratively localize a feasible solution. At their core, those methods are an elegant and efficient way to greedily solve localization problems in a manner that is resembling a lot to some machine learning procedures, particularly the Perceptron, in the sense that their general scheme is to start from an arbitrary tentative solution and iteratively refine it based on data feedback.

Algorithm 5 depicts a skeletal example of Cutting Planes algorithm that underlines the general idea while leaving the more specific parts of the algorithm unspecified.

Cutting Planes methods are built upon the idea that localization problem, however complex they are, verify the two following observations:

- A lot of elements in $\mathbf{X}$ are actually redundant, that is to say, there is a lot of hyperplanes $h_{\mathbf{X}_i}$ such that $\mathcal{W}(\mathbf{X} \backslash \{h_{\mathbf{X}_i}\}) = \mathcal{W}(\mathbf{X})$.

**Algorithm 5** A Skeletal Cutting Planes Algorithm for finding a point in $\mathcal{W}$

**Require:** A Cutting Planes Oracle $O$
**Ensure:** A point $\boldsymbol{w}^k \in \mathcal{W}$
1: $\mathcal{C}^0 \leftarrow \mathcal{B}_1$
2: $k \leftarrow 0$
3: **repeat**
4: $\quad \boldsymbol{w}^k \leftarrow \textsc{Query}(\mathcal{C}^k)$ ▷ Compute a *query point* $\boldsymbol{w}^k \in \mathcal{C}^k$
5: $\quad$ Ask the Oracle $O$ whether $\boldsymbol{w}^k \in \mathcal{W}$
6: $\quad$ **if** $\boldsymbol{w}^k \notin \mathcal{W}$ **then**
7: $\quad\quad$ Receive a *Cutting Plane* $h_{\boldsymbol{c}}$ from $O$
8: $\quad\quad \mathcal{C}^{k+1} \leftarrow \mathcal{C}^k \cap \{\boldsymbol{w} \in \mathbb{X} | \langle \boldsymbol{c}; \boldsymbol{w} \rangle \geq 0\}$ ▷ Trim $\mathcal{C}^k$
9: $\quad\quad k \leftarrow k + 1$
10: $\quad$ **end if**
11: **until** $\boldsymbol{w}^k \in \mathcal{W}$
12: **return** $\boldsymbol{w}^k$

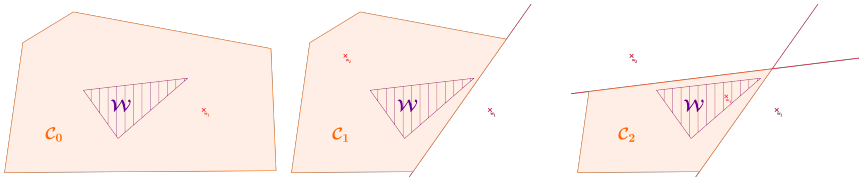



Figure 5.3 – *A schematic representation of how Cutting Planes methods proceed. The purple area represents the version space and the orange one the search space. Each panel corresponds to a different query (the red dot) after which the oracle return a separating cut, hence reducing the size of the search space.*

- Finding a point $\boldsymbol{w} \in \mathcal{W}$ up front may be too difficult to tackle as a single task.

Typically, a Cutting Planes procedure may start with a very simple localization problem, involving only a few hyperplanes and then iteratively refine the problem by adding new hyperplanes until a point in the version space is found.

Cutting Planes algorithm are built upon the idea of Cutting Planes oracle —denoted $O$ henceforth— where $O$ is a function that, given a point $\boldsymbol{w} \in \mathbb{X}$ answers `yes` or `no` whether $\boldsymbol{w} \in \mathcal{W}$; in addition, if the oracle answers negatively, it responds with a separating cut, that is a hyperplane's normal vector $\boldsymbol{x}$ such that $\langle \boldsymbol{w}; \boldsymbol{x} \rangle < 0$; in other words $h_{\boldsymbol{x}}$ separates $\boldsymbol{w}$ from $\mathcal{W}$. Assuming it has access to a Cutting Planes oracle $O$, the algorithm then keeps track of a *search-space* $\mathcal{C} \supseteq \mathcal{W}$ that is iteratively trimmed until it is close enough to $\mathcal{W}$. Therefore the algorithm works as follows: it first computes a *query point* $\boldsymbol{w} \in \mathcal{C}$ and asks the oracle whether $\boldsymbol{w} \in \mathcal{W}$. Then, it updates $\mathcal{C}$ with respect to the oracle answer, or terminates if $\boldsymbol{w} \in \mathcal{W}$.

To complete this section note that Cutting Planes algorithms get their name from their general query scheme that consists in starting with a very general search space $\mathcal{C}$ and iteratively shrink it until $\mathcal{C}$ becomes an acceptable approximation of $\mathcal{W}$. More precisely for a Cutting Planes algorithm to terminate $\mathcal{C}$ does not have to be exactly the same as $\mathcal{W}$. As long as the algorithm is fully deterministic —especially the computation of the *query point* and the oracle answer— it is guaranteed that the last

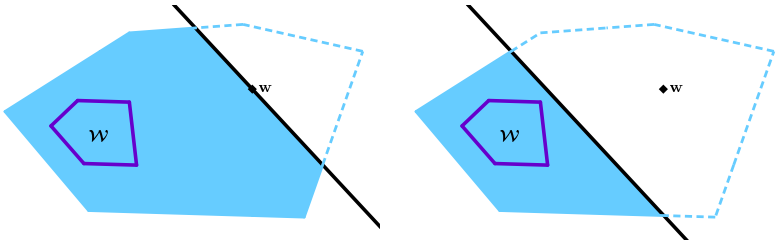



Figure 5.4 – *An illustration of neutral (left) and deep (right) cuts. The search space is represented in blue and the version space is the purple pentagon. The query point and the Oracle's cut are in black. In both figure, the dotted area represents the portion of the search space that is pruned by the cut.*

query will always be for a point in $\mathcal{W}$ given $\mathcal{C}$ is the same. This allows in some case to greatly reduce the complexity of $\mathcal{W}$ by rebuilding a neat version of it through $\mathcal{C}$. Therefore, if $\mathbf{X}$ contains redundant information, Cutting Planes algorithms may identify a small set of cuts that describe an accurate approximation of $\mathcal{W}$. In other words, they encompass a very desirable feature for learning algorithm in that they naturally provide a Sample Compression Scheme. In anticipation to the discussions to come in chapter 6, we may already mention that this property will be most desirable when the aforementioned cuts will be drawn from $\mathbf{X}$.

### 5.2.2 Analysis of Cutting Planes Algorithms through the query step

The *query step* is the cornerstone of Cutting Planes algorithms. It is this step that drives the execution of the algorithm, and ultimately determines whether a particular instance of the algorithm will be fast and/or efficient. In a nutshell, the *query step* is divided two folds: first, the QUERY function compute a new query point $\boldsymbol{w} \in \mathcal{C}$ (see Alg. 5); then $\boldsymbol{w}$ is fed to the Oracle and a cut is received.

#### The role of the Oracle

By design, the Oracle acts as a black box of which we do not have any knowledge. As such, no assumption should be made on its internal mechanism beyond the fact that it provides feedback through cuts in a deterministic way. Hence, the Oracle drives the cutting process by providing separating hyperplanes when needed —where, contrary to learning, separating here relates to the query point and $\mathcal{W}$.

Building on this idea, we may differentiate two types of cuts referred as *deep* and *neutral*. A neutral cut is a cut that goes through the query point in such a way that $\boldsymbol{w}$ lies just outside the edges of the updated search space. On the other hand, a deep cut refers to a cut that truly goes *in between* $\boldsymbol{w}$ and $\mathcal{W}$ in the sense that there is some strictly positive distance between $\boldsymbol{w}$ and the cut (see also Fig 5.4). Intuitively, a deep cut is always better than a neutral one because it helps shrinking $\mathcal{C}$ faster although our lack of knowledge about the oracle does not permit to plan for a deep or a neutral cut during the querying process and neutral cuts should always be assumed as far algorithm analysis goes.

#### The role of the query point

As a matter of fact, the QUERY() function amounts to solve a localization problem per se. However, the search space has some nice properties in this respect. First because it is defined by much fewer hyperplanes than the version space —one per step of the algorithm. And second, because each update only adds a single constraint at a time, thus at a given step $k$, $\mathcal{C}^k$ is supposedly close to $\mathcal{C}^{k-1}$ which in turns implies that $w^k$ should not be too far away from $\boldsymbol{w}^{k-1}$ although this last bit is not forcefully accurate, especially with respect to what 'far away' might means. However the intuitive idea it conveys is that the sub-problem solved at each iteration should not be thought as independent localization tasks but steps on the path to the final solution where each update of $\boldsymbol{w}$ is a little closer to $\mathcal{W}$.

In this regard, query points play a central role in Cutting Planes algorithm and the general efficacy of a particular implementation is often tied to how well query point computation and Oracle's cuts intertwine together. On the other hand, it is a core idea that computing a query point should be a simple process or otherwise it would undo the purpose of Cutting Planes methods. In the end, these considerations call for carefully designed query point strategies that we may precise in the remaining of this chapter.

### 5.2.3 Toward Efficient Cutting Planes Algorithms

As stated previously, designing a query strategy is instrumental in ensuring the efficacy of a Cutting Planes method. The main challenge here is to find, in a reasonable time, a query point which will come with guarantees on the reduction of $\mathcal{C}$, independently of the cut returned by the Oracle.

Intuitively, we want query points that are deep within $\mathcal{C}$ so cutting them out will lead to a drastic reduction in the size of the search space. For instance, consider two query points $\boldsymbol{w}$ and $\boldsymbol{w}'$ such that $\boldsymbol{w}$ is *deep* within $\mathcal{C}$ and $\boldsymbol{w}'$ lies near an edge of $\mathcal{C}$. For now, it is unnecessary to detail what *deep* might mean in the mathematical sense, so let it just be a generic term for *'away from all edges'* and we will refine the idea later. Now, for the case of $\boldsymbol{w}$, no matter the cut, we are more or less ensured that any cut that leaves out $\boldsymbol{w}$ will also prune a good chunk of $\mathcal{C}$. However, in the case of $\boldsymbol{w}'$ there is two possibilities: either the cut removes only a small part of $\mathcal{C}$ or $\mathcal{W}$ is lies between $\boldsymbol{w}'$ and its closest $\mathcal{C}$'s edge, thus leading to a cut that will discard most of $\mathcal{C}$ (see also fig 5.5)

Obviously, this does not mean that querying points close to the edges of $\mathcal{C}$ is desirable, because of our lack of control over the oracle, the preferred approach is to ensure a guaranteed minimum size reduction, assuming that the oracle will always return the least favourable cut. Hence, the query point should ensure a large enough reduction of $\mathcal{C}$'s size in all cases.

We have yet to come with a mathematical characterization of what a good query point is beyond the, rough, *'away from all edges'* placeholder we have used for now and the intuition that this term conveys is the one of the center of $\mathcal{C}$, which will be the central topic of the next section.

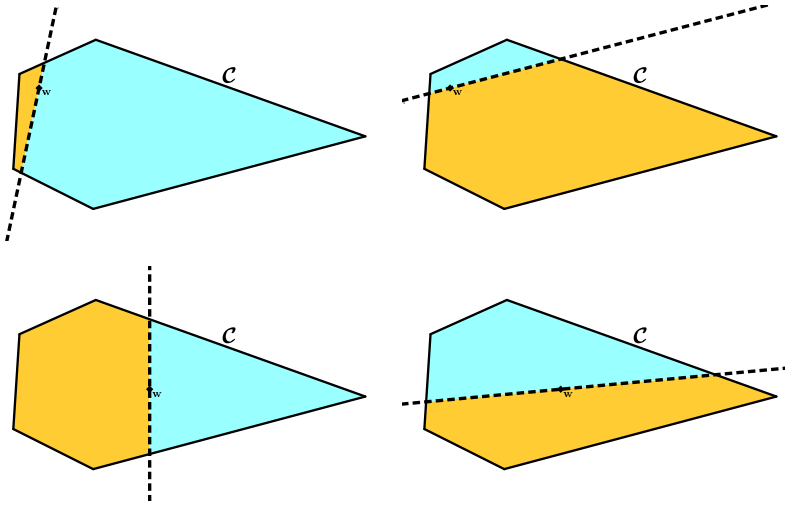



Figure 5.5 – *Two different query points and the cuts they may induce; the orange (resp. blue) area represents the discarded (resp. kept) part of $\mathcal{C}$ after the cut. The two top figures depict the case of a query point close to the edges of $\mathcal{C}$ and the induced cuts split $\mathcal{C}$ unevenly, leading to inconsistent reduction in the volume of $\mathcal{C}$. On the other hand, the bottom figures depict the case of a* central *query point. In this case the split is almost even in all cases and leads to a much more reliable reduction of $\mathcal{C}$'s volume.*

## 5.3 Centroids

### 5.3.1 The epitome of centroid: the Center of Gravity

In the previous section, we haven taken some time to emphasize the stakes of a good query strategy for Cutting Planes methods. Unsurprisingly, studying those strategies has been a major focus of Cutting Planes related research since the inception of the algorithm (see e.g. [Boyd and Vandenberghe, 2004] for a general review). The idea of a point *'away from all edges'* matches the mathematical concept of *centroid*, although there exists many different kinds of centroid, with different properties, more or less desirable. Conceptually, a centroid is defined as the *center* of a convex body, where the term *'center'* is up to different meanings, hence the multiple co-existing variety of centroids.

#### A definition of the Center of Gravity

The first centroid to be considered for query purposes in the Cutting Planes literature is also the most intuitive one: the center of gravity, or alternatively, center of mass. The term is borrowed from physics terminology and it actually correspond to the usual center of mass of an object, that is the average location of an object[1].

**Definition 5.1** (Center of Gravity) *Given a convex body $\mathcal{C}$ assumed to be of constant density, the center of gravity of $\mathcal{C}$ denoted $CG(\mathcal{C})$ is:*

$$CG(\mathcal{C}) \doteq \frac{1}{Vol(\mathcal{C})} \int_{\boldsymbol{w} \in \mathcal{C}} \boldsymbol{w} d\boldsymbol{w}$$

[1]For a more complete definition, we should take into account the mass distribution over said object. Although, in an effort to keep things simple, we assume an uniformly distributed mass.

where $\text{Vol}(\mathcal{C})$ is the volume of $\mathcal{C}$.

**Definition 5.2** (Volume of a convex body) *Given a convex body $\mathcal{C}$, its volume $Vol(\mathcal{C})$ is defined as*

$$Vol(\mathcal{C}) \doteq \int_{\boldsymbol{w}\in\mathcal{C}} 1 d\boldsymbol{w}$$

Centers of gravity are remarkable in many senses, they are easy to grasp and understand, yet they crystallize some of the most fundamental properties of a convex body. Considering centers of gravity as potential query point for Cutting Planes algorithms started in the early 50's with the works of [Newman, 1965, Levin, 1965] (see also [Boyd and Vandenberghe, 2007]).

### The fundamental property of Center of Gravity

What makes Center of Gravity interesting as query points is that they match the very idea of *center* in a Cutting Planes sense. That is, for a given convex search space $\mathcal{C}$, its center of gravity is the point that have the best volume reduction guarantees if cut-out by the oracle. This property is often referred as the *fundamental* property of Centers of gravity and is due to [Grunbaum, 1960, Levin, 1965]

**Theorem 5.1** (Partition of Convex Body) *Let $\mathcal{C}$ be a convex body in a D-dimensional Hilbert space, $CG(\mathcal{C})$ its center of gravity. Let $h_{\boldsymbol{x}}$ any hyperplane of normal vector $x$ that splits $\mathcal{C}$ in two partitions $\mathcal{C}^+$ and $\mathcal{C}^-$ such that*

$$\mathcal{C}^+ \doteq \{\boldsymbol{w} \in \mathcal{C} : \langle \boldsymbol{w}; \boldsymbol{x}\rangle \geq 0\}$$
$$\mathcal{C}^- \doteq \{\boldsymbol{w} \in \mathcal{C} : \langle \boldsymbol{w}; \boldsymbol{x}\rangle < 0\}$$

*If $CG(\mathcal{C}) \in \mathcal{C}^-$ (resp. $\mathcal{C}^+$) then*

$$Vol(\mathcal{C}^-) \geq \left(\frac{D}{D+1}\right)^D Vol(\mathcal{C})$$

Because the term $\left(\frac{D}{D+1}\right)^D$ decreases as $D$ increases and converge to $e^{-1}$, we also have the following corollary that is interesting because the result no longer depends on $D$.

**Corollary 5.1** (Partition of Convex Body (continued)) *Let $\mathcal{C}$, $\mathcal{C}^-$, $\mathcal{C}^+$ and $CG(\mathcal{C})$ defined as in theorem 5.1 then, if $CG(\mathcal{C}) \in \mathcal{C}^-$ (resp. $\mathcal{C}^+$):*

$$Vol(\mathcal{C}^-) \geq e^{-1} Vol(\mathcal{C}) \approx 0.37 Vol(\mathcal{C})$$

A detailed rewriting of the theorem's proof can be found in Appendix A.3. Also, a novel, non-trivial extension of this theorem can be found in Section 5.3.4 theorem 5.2.

This theorem is remarkable on many levels. First it holds for any convex body, independently of their shape on complexity [2] and the corollary makes it independent to the dimension which is a very unusual and remarkable feature for such results. Also, it establishes a strong link between the volume of a convex body and its center of gravity, and, in a

[2] Also, note that the result holds when hyperplanes are defined with an offset term

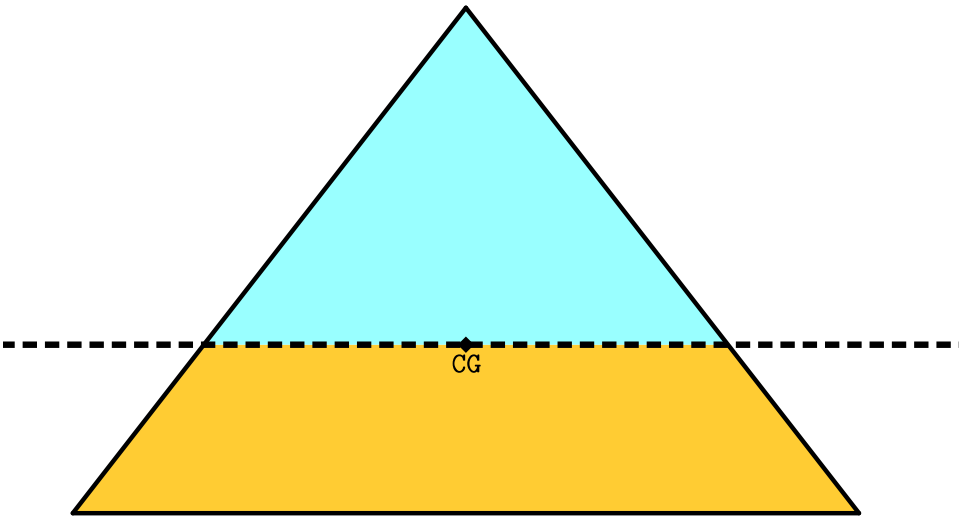



Figure 5.6 – *The limit case of Theorem 5.1 in two dimensions. The center of gravity is located precisely at* $2/3$ *of the total height and the blue area amounts for exactly* $4/9$ *of the total area of the triangle. The proof of the theorem is based on a generalization of this case to any dimension and can be found in full in appendix A.3.*

sense, the center of gravity encodes some fundamental volume and shape information; on that matter, it thus comes with no surprise that computing the center of gravity allows for fast computation of the volume (see [Rademacher, 2007, Elbassioni and Tiwary, 2008]). Finally, it establishes a geometric convergence rate for Cutting Planes algorithms when centers of gravity are used for in the query strategy.

More precisely, we know that given an instance of Cutting Planes algorithm, the execution stops when $\text{Vol}(\mathcal{C})$ is close to $\text{Vol}(\mathcal{W})$ as it means that $\mathcal{C}$ is a close approximation of $\mathcal{W}$ and consequently localizing a point in $\mathcal{C}$ is to localize a point in $\mathcal{W}$. The limit case being when the two volumes are equal, since $\mathcal{C} \supseteq \mathcal{W}$ it implies that $\mathcal{C} = \mathcal{W}$. Therefore, theorem [Grunbaum, 1960] allows to upper bound the number of steps performed by a Cutting Planes algorithm that would use $CG(\mathcal{C})$ as a query point. More precisely we have that the volume of $\mathcal{C}$ shrinks by a constant ratio at each step, hence the number of step before the algorithm terminates, that is the number of iteration before $\text{Vol}(\mathcal{C}) \leq \text{Vol}(\mathcal{W})$, is in $\mathcal{O}\left(\log(\text{Vol}(\mathcal{W}))\right)$.

**Computational limits of the center of gravity**

For all their nice properties, Centers of gravity nonetheless suffer from a major drawback: they are difficult to compute. More precisely, because of their close relation with the volume of a convex body, Centers of gravity can be used to iteratively compute the volume of a convex body. Because volume computation is a known #P-hard problem, computing a center of gravity is therefore also #P-hard —See [Rademacher, 2007, Elbassioni and Tiwary, 2008] for more details.

With exact computation of centers of gravity not a possibility, one is left with approximation methods. We shall review some of those later

in Section 5.3.3 in order to keep with our main focus here, that is the exposition of different types of centroid.

### 5.3.2 A second centroid, the chebyshev's center

Motivated by the hardness of computing centers of gravity, *Chebyshev's Centers* were the next step in terms of centroid in the Cutting Planes literature [Elzinga and Moore, 1975]. They can be thought as a relaxation of centers of gravity in that they try to capture the same concept of *center* though with a far simpler —in the computational sense— definition [3].

Intuitively, the Chebyshev's Center of a convex body is the point that is *the farthest from all edges* which seems to fit our idea of an ideal query point. More formally, given a convex body $\mathcal{C}$ it's Chebyshev's Center is the center of the largest inscribed sphere in $\mathcal{C}$, hence the idea of being far from all edges.

**Definition 5.3** (Chebyshev's center) *For a given convex body $\mathcal{C}$ we define the Chebyshev's center of $\mathcal{C}$ and write $CC(\mathcal{C})$ the point such that:*

$$CC(\mathcal{C}) \doteq \min_{\boldsymbol{w}\in\mathcal{C}} \max_{\boldsymbol{w}'\in\mathbb{X}\setminus\mathcal{C}} \{\|\boldsymbol{w}-\boldsymbol{w}'\|_2\}$$

In other words, $CC(\mathcal{C})$ is the deepest point in $\mathcal{C}$ in the sense that it is far away from all edges. Nonetheless $CC(\mathcal{C})$ is only an approximation of the center of gravity ([Tong and Koller, 2001, Herbrich et al., 2001, Boyd and Vandenberghe, 2007, Boyd and Vandenberghe, 2004]), underlying that the problem of finding the optimal query point is more complex that one might thought at first.

**Example 5.1** (Trapezoid) *A good example of how Chebyshev and gravity centers relate to each other is depicted in figure 5.8. Consider a trapezoid defined as the resulting intersection of a triangle and a half-space with delimiting hyperplane parallel to the triangle's base. For the sake of the argument, let assume that the height of the trapezoid is long enough so that the Chebyshev centers of the triangle and the trapezoid are the same, that is, the intersection of the two angle bisectors from the base (see fig 5.7).*

*However, pruning the upper angle of the triangle does impact the center of gravity that is consequently shifted toward the base of the trapezoid (Fig. 5.8). Hence, we can define a whole variety of trapezoid from this construction with different heights; each one of those have the same Chebyshev's center but different centers of gravity. The two limit cases being, on the one hand, the full triangle (Fig. 5.8 bottom panel), and on the other hand the trapezoid with height equal to the diameter of the maximal radius inscribed sphere (Fig. 5.8 top panel).*

Another example is also depicted in figure 5.9, which represents an actual version space restricted to the unitary Euclidian ball in a three dimensional learning problem along with its Chebyshev's center (dark blue) and center of gravity (light blue).

The key observation is that the edges of a convex body does not capture enough information to properly assert its mass repartition (see the

[3] It seems to exist two non-equivalent definitions of the Chebyshev's Center. The one we will use is the one of —among others— [Boyd and Vandenberghe, 2007]. The other one is, for instance, the one described in [Garkavi, 1964, Amir and Mach, 1984]

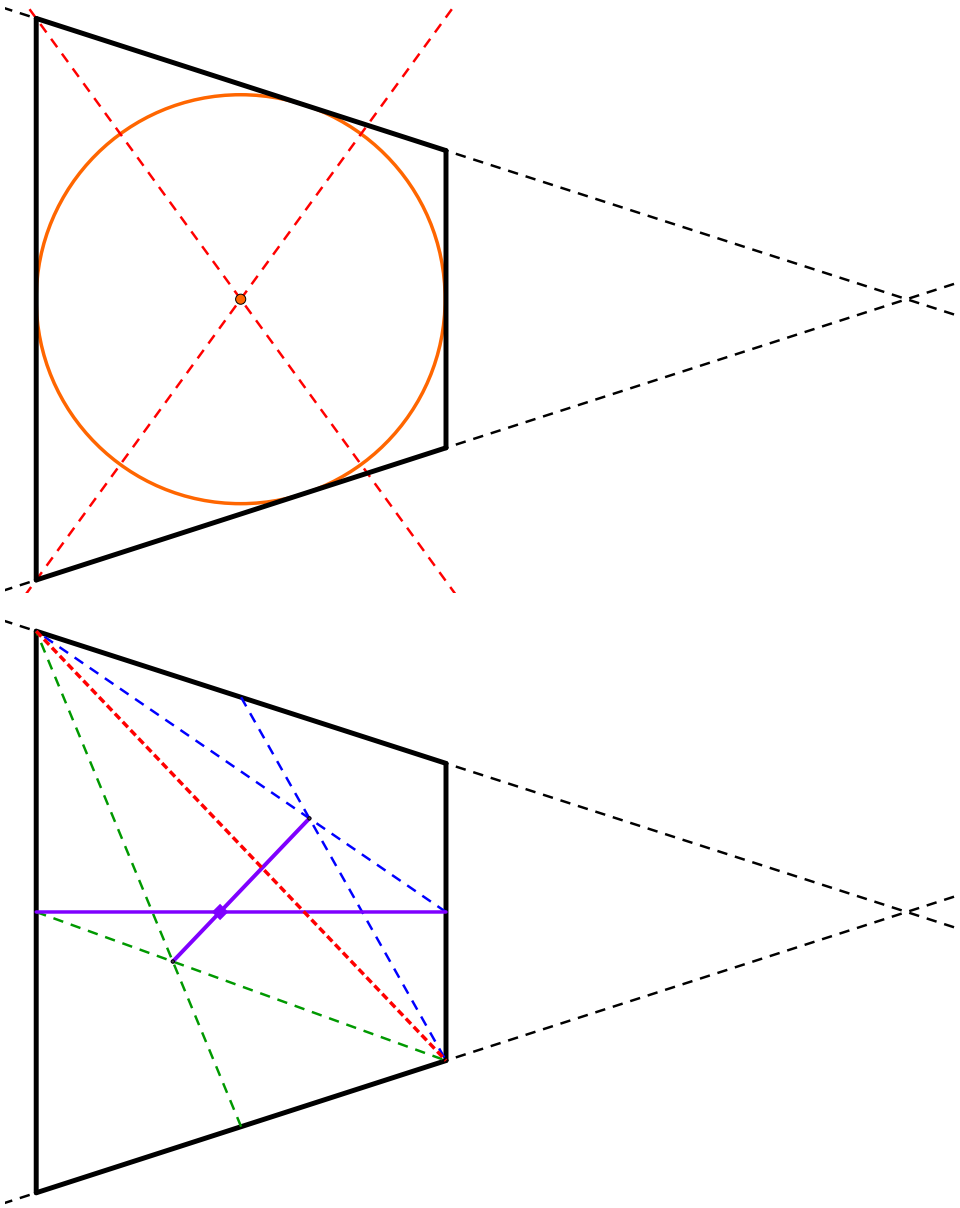


Figure 5.7 – *An illustration of the Chebyshev's center (top) and the center of gravity (bottom) for a trapezoid. Geometrically, the Chebyshev's center can be defined as the intersection of the two base's angle bisector (red lines in the top figure) —given a long enough height. The center of gravity case is a little more convoluted as it requires to first split the trapezoid in two triangles (red line). From here, we can easily find the center of gravity of the two sub-triangles from their medians (green and blue lines). Finally, we know that the Trapezoid's center of gravity lies both between the two sub-triangle center of gravity —remind that it can be defined as the weighted average of those two points—* and *on the trapezoid's base median. Taking intersection of those two lines (purple) yields the desired point.*

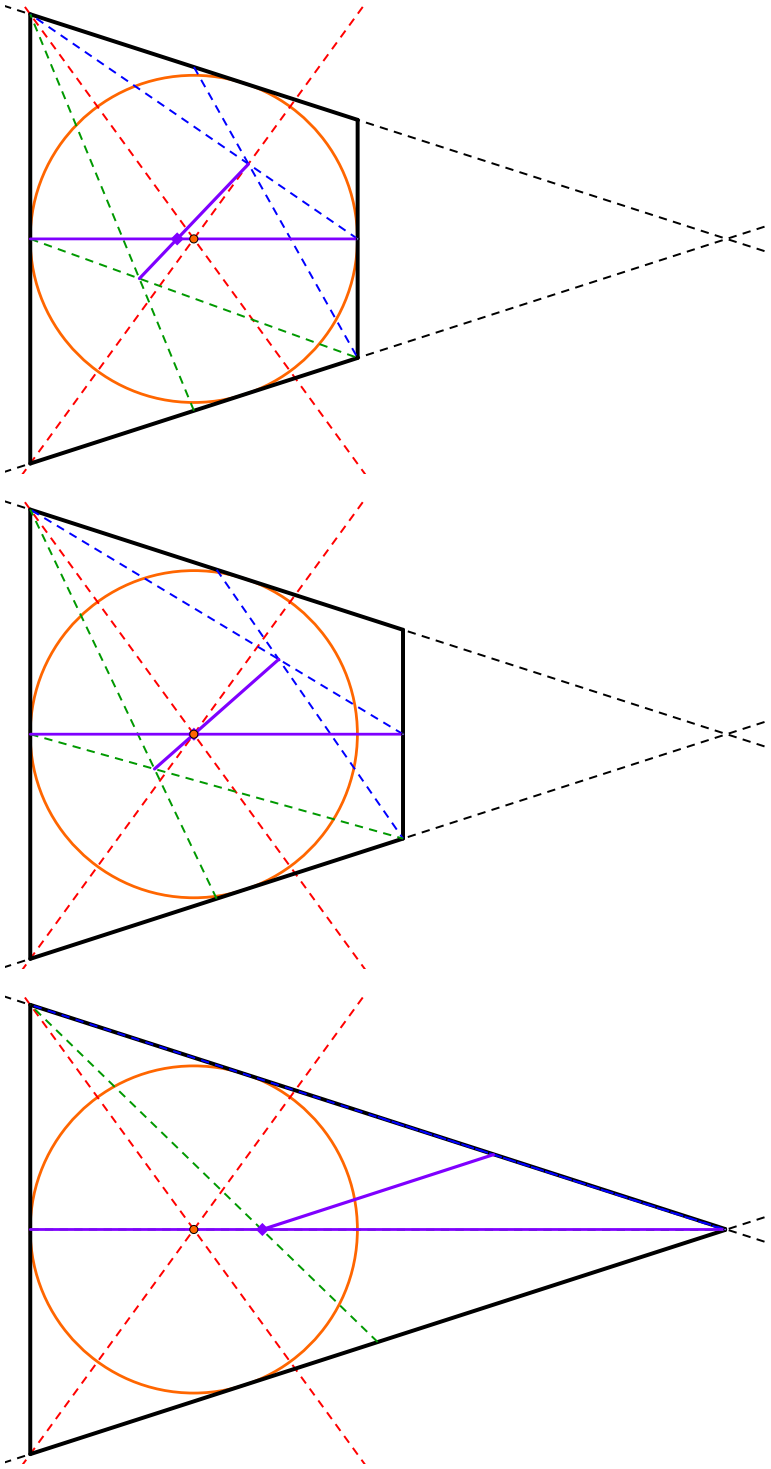


Figure 5.8 – *Three different trapezoids with three different heights. The top and bottom cases correspond to the upper and lower limits for the height. That is when the height is equal to the diameter of the largest inscribed sphere (orange, top panel) and the triangle case (bottom panel). The middle panel depicts the case where the Chebyshev center and the center of gravity are the same point. Some of the construction lines of Fig. 5.7 are depicted for reference.*

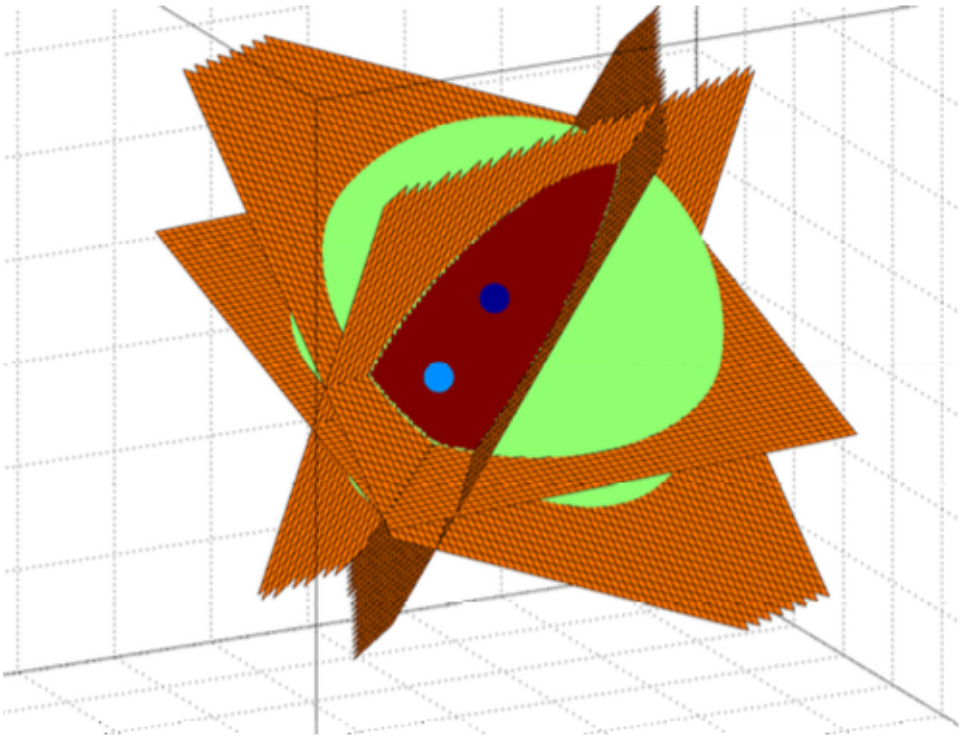


Figure 5.9 – *A depiction of the difference between the center of gravity and the Chebyshev's center on a 3-dimensional search space (red area) lying on the unit ball of norm one. The orange hyperplanes represent cuts. The light blue (resp. dark blue) point is the Chebyshev's center (resp. center of gravity). We can see here that the Chebyshev's center is drastically shifted with respect to the center of gravity.*

discussion in [Tong and Koller, 2001] for instance). This eventually explain the lack of theoretical guarantees related to Chebyshev's center when used in conjunction with Cutting Planes methods. Contrary to Centers of Gravity, there is no strong convergence results for Chebyshev's centers and their use in practice is more backed by empirical efficiency rather than theoretical analysis.

On the other hand, Chebyshev's centers are easy to compute and, given a convex body described as a finite set of ellipsoid constraints —which is a more general setting than needed for our case— one can find it by simple quadratic programming.

Finally, on the chapter of approximate center of gravity, we should also mention briefly another popular approximation, that is the analytical center —ACCPM— [Nesterov, 1995, Boyd and Vandenberghe, 2007] which can be thought as a middle ground between the guarantees of a center of gravity, and the computational ease of the Chebyshev's Center. Moreover, ACCPM has recently found its way to the machine learning literature [Fanzi et al., 2009]. Nonetheless, we will not go this way in this thesis and investigating the use of ACCPM in machine learning is yet another possible, and promising, extension of our work.

### 5.3.3 Sampling methods

Another take on the problem of computing the center of gravity is that of seeking an analytical approximation of the centroid —as opposed to the geometrical approximation that is the Chebyshev's Center. In other word, the idea is to compute a point that is relatively close —from a distance perspective— to the center of gravity, yet easy to compute.

#### MCMC methods

This can be achieved by a simple *Monte Carlo* approach. Namely, we will approximate the center of gravity by summing over a finite sample of points uniformly spread over $\mathcal{C}$. Let consider $\mathscr{U}_{\mathcal{C}}$ the uniform distribution over $\mathcal{C}$ such that, for all $\boldsymbol{x} \in \mathbb{X}$:

$$\mathscr{U}_{\mathcal{C}}(\boldsymbol{x}) = \begin{cases} \frac{1}{\text{Vol}(\mathcal{C})} & \text{if } \boldsymbol{x} \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

and a collection $\mathcal{E} = \{\boldsymbol{w}_1, \dots, \boldsymbol{w}_k\}$, $\mathcal{E} \subset \mathcal{C}$ of $k$ points drawn from $\mathscr{U}_{\mathcal{C}}$: $\mathcal{E} \overset{\text{i.i.d}}{\sim} \mathscr{U}_{\mathcal{C}}^k$. Then, we define the *approximate center of gravity* $\widetilde{CG}$ as:

$$\widetilde{CG} \doteq \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{w}_i$$

However, sampling uniformly over $\mathcal{C}$ is not a trivial task. Fortunately, the question of sampling points from a given distribution is a vastly studied topic and methods abounds to achieve this goal. We are particularly interested in *Monte-Carlo Markov-Chain* methods —or MCMC for short— which rely on biased random walks over the distribution's domain. There exists a wide range of MCMC methods, and

going through a review of the MCMC literature in this work is unrealistic and unwise. Hence, we will focus in the following on a peculiar family of MCMC sampling method, known as *Hit and Run algorithms* [Lovász, 1999, Lovász and Vempala, 2003] that is famously known as a good sampling algorithm over convex bodies —that is, when the target distribution is constant over the convex body, and 0 elsewhere. In its simplest form, the Hit-and-Run algorithm starts from a point within $\mathcal{C}$ and perform a random walk within the boundaries of $\mathcal{C}$. Notably, Hit-and-Run algorithms converges to the uniform distribution in a time polynomial in the dimensionality of $\mathcal{C}$ [Lovász, 1999] and in practice happens to be one of the fastest class of sampling algorithms.

**The billiard Algorithm**

For mainly practical reasons, we will consider a slight variation of the original Hit-and-Run algorithm that is known as the *billiard algorithm* [Rujan, 1997, Rujan and Marchand, 1999]. Historically this variation has been reliably used in center of gravity focused work and machine learning [Herbrich et al., 2001] hence this choice. The idea behind the billiard algorithm is a very simple one, like regular Hit-and-Run, the algorithm starts with an initial point within $\mathcal{C}$ and move in a random direction. However, the algorithm keeps moving until it reaches an edge of $\mathcal{C}$ and then bounce on this boundary as a billiard ball would. Finally, sampling the trajectory at a constant time interval provides the required sample set. The algorithm convergence rests on the ergodic billiard theory which studies and describes the behaviour of a billiard ball undergoing elastic collision — that is, without loss of velocity— with the (enclosing) edges of a convex body. Roughly speaking, ergodicity ensure that almost all infinitely long trajectories will cover uniformly the phase space. In other words, assuming an infinitely long trajectory, the billiard's ball will uniformly assume all the possible combinations of location and direction within $\mathcal{C}$ eventually. The question of what conditions are needed for the version space to have this ergodicity property is, to the best of our knowledge, still open. However in his work, Pál Ruján [Rujan, 1997] strongly implies any version space can reasonably be considered as ergodic:

"Except for very special cases with high symmetry, it seems therefore unlikely that high dimensional convex polyhedra are not ergodic. If so, adding a few scatterers inside the polyhedra might restore ergodicity."

Moreover, this is an assumption that has been backed by numerous empirical results [Rujan, 1997, Herbrich et al., 2001, Rujan and Marchand, 1999, Brinker, 2004].

In practice, running the billiard algorithm amounts to compute, repeatedly which edge of $\mathcal{C}$ the ball will hit first. That is to say, given a search space defined as the intersection of a collection of cuts of normal vectors $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M$, and a point $\boldsymbol{w} \in \mathcal{C}$ moving at constant speed along a random direction $\boldsymbol{v}$ we have to compute for every delimiting hyperplane $h_{\boldsymbol{c}_i}$ of $\mathcal{C}$ how long it will take for $\boldsymbol{w}$ to collide with $h_{\boldsymbol{c}_i}$. We note $\text{dist}(\boldsymbol{w}, h_{\boldsymbol{c}_i})$ the distance between $\boldsymbol{w}$ and $h_{\boldsymbol{c}_i}$ and $\boldsymbol{v}_i$ the component of $\boldsymbol{v}$ perpendicular

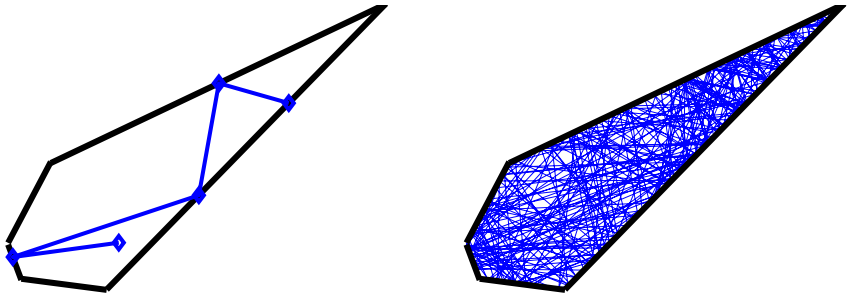


Figure 5.10 – *An example run of the billiard algorithm on a 2 dimensional convex polytope. The left figure depicts the five first bounces whereas the right one is a plot of the billiard's trajectory after 300 bounces.*

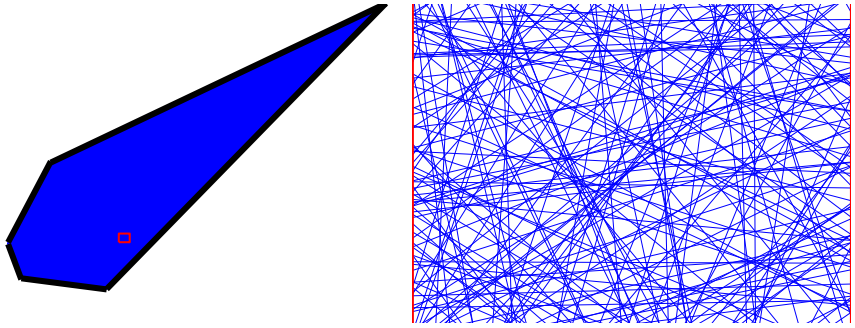


Figure 5.11 – *A depiction of the ergodic nature of the billiard trajectory. The left figure represents the complete billiard path after a typical run of 5000 bounces. The left figure is a magnification of the left panel's red square. Although the polytope seems to be uniformly covered in the first figure, we can distinguish some irregularities at a lower scale.*

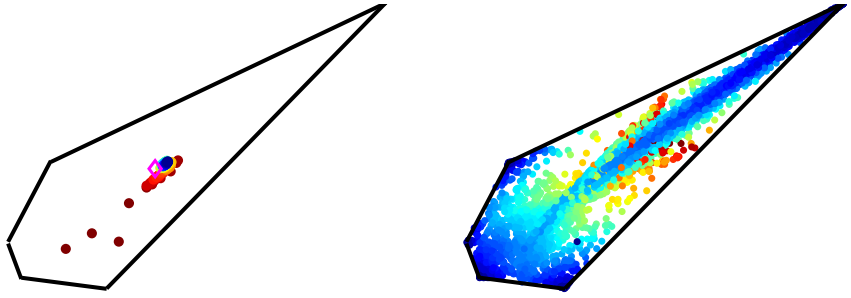


Figure 5.12 – *An illustration of the billiard algorithm used to approximate the center of gravity. The left figure depicts the first hundred approximation made after the first hundred of bounces. The color correspond to the order, with the red one being the first approximation, and the blue one the hundredth one. The purple diamond represent the final estimate after 5000 bounces. Notably, even after a hundred of bounce, the algorithm already yield a relatively good approximation. The right picture represents the midpoints of each of the 5000 bounces, colored by the length of the bounce. The final approximation of the center of gravity is then computed from the weighted average of those points. Contrary to the billiard's trajectory, those points do not form an uniform coverage of the polytope.*

to $\boldsymbol{c}_i$. Hence

$$\operatorname{dist}(\boldsymbol{w}, h_{\boldsymbol{c}_i}) = \langle \boldsymbol{w}; \boldsymbol{c}_i \rangle \tag{5.4}$$

$$\boldsymbol{v}_i = \langle \boldsymbol{v}; \boldsymbol{c}_i \rangle \tag{5.5}$$

and the time before $\boldsymbol{w}$ collides with $h_{\boldsymbol{c}_i}$ is therefore:

$$\tau_i = -\frac{\operatorname{dist}(\boldsymbol{w}, h_{\boldsymbol{c}_i})}{\boldsymbol{v}_i}$$

After computing the $M$ collision times, it suffices to looks for the smallest positive one $\tau_{\min} = \min_i \tau_i$ to determines which hyperplane will intercept $\boldsymbol{w}$ first. A special case occurs when the billiard leaves the ball of unit norm $\mathcal{B}_1$ before hitting any hyperplane. Some of the aforementioned works propose to reset the billiard when this event happens [Rujan, 1997, Herbrich et al., 2001]. In our case, it seems justified to treat this case as a regular collision [Minka, 2001b, Brinker, 2004] and stay as close as possible to the original billiard setting. This also make sense from the geodesic point of view since the center of gravity of $\mathcal{C}$, when projected on the surface of $\mathcal{B}_1$ matches the center of gravity of $\mathcal{C} \cap \mathcal{B}_1$ (see section 5.1.1). Practically, detecting collision with the inner limit of $\mathcal{B}_1$ is a matter a fixing a maximum collision time $\tau_{\max}$ after which $\|\boldsymbol{w}\tau_{\max}\boldsymbol{v}\| = 1$; this is simply done by solving a quadratic equation after each bounce.

Finally, after a collision, the new velocity vector is given by

$$\boldsymbol{v}' = \boldsymbol{v} - 2\boldsymbol{v}_i\boldsymbol{c}_i$$

Additionally, assuming a normal vector $\boldsymbol{c}_i$ of unit norm $\|\boldsymbol{x}_i\| = 1$ we can see that the velocity is preserved —that is, $\|\boldsymbol{v}'\| = \|\boldsymbol{v}\|$— and $\langle \boldsymbol{v}'; \boldsymbol{c}_i \rangle = -\langle \boldsymbol{v}; \boldsymbol{c}_i \rangle$. Note that, when the collision involves the unit ball $\mathcal{B}_1$, we assume $\boldsymbol{x}_1$ to be a vector located at the impact point and directed toward the origin of the space.

One practical advantage of the billiard algorithm is that both the collision time and velocity update are already defined in term of dot product. Thus it simplifies a lot the process of rewriting the algorithm for kernelized data and classifier; notably, this is one of the motivation of [Herbrich et al., 2001] for using billiard algorithm. Moreover, dealing with kernels incidentally means that generating a random direction uniformly over the unit sphere in $\mathbb{F}$ is not a trivial task [Gilad-Bachrach and Burges, 2013]. Contrary to the vanilla Hit-and-Run algorithm, being able to do so is not a prerequisite for the billiard algorithm due to the ergodicity property which states that almost all trajectory will eventually cover the entire state space.

Finally, note that the above discussion is obviously subject to our capacity of finding a good starting point, that is a point in $\mathcal{C}$. We already argued that this is a localization problem supposedly easy to tackle, nonetheless, the biggest computational burden comes in practice from this initialization step. On the other hand, because the billiard algorithm visits (hopefully) the totality of $\mathcal{C}$, one can adopt a practical approach and consider the sampled points of $\mathcal{C}$ as potential starting point for the next iteration —remember that the first iteration is trivial since $\mathcal{C}_0 = \mathcal{B}_1$. If all else fail, a viable fall back strategy is to use a fast machine learning algorithm (e.g. the perceptron) to quickly find a point in $\mathcal{C}$.

### 5.3.4 On the property of sampled centers of gravity

#### The problem so far

From an empirical perspective, sampling methods provide useful results that are practically close to the true center of gravity and, more importantly, Theorem 5.1 seems to hold for these approximates [Rujan, 1997, Herbrich et al., 2001]. Theoretically though, a formal result is lacking for numerical estimate of the center of gravity. More precisely, to the best of our knowledge, there is no previous result on how the bound of 5.1 degrades when considering a close estimate of the center of gravity. Hopefully, the ratio of theorem 5.1 should degenerate smoothly with the quality of the approximation, that is its distance from the true center of gravity. One of the contribution of this thesis is to answer this question and provide a generalization of 5.1 that holds for any approximation of the center of gravity.

Theorem 5.2 extends Theorem 5.1 for approximate centers of gravity. In addition it reduces to Theorem 5.1 when applied to the true center of gravity. Moreover, that is a result of its own interest, which may benefit to many fields of computer science. A prime example being sampling methods as previously discussed, and of course Cutting Planes methods. Here, the purpose of Theorem 5.2 is essentially to the above case, that is to validate the use of approximations of the center of gravity $CG(\mathcal{C})$ in a Cutting Planes setting.

#### A generalized partition of convex bodies

**Theorem 5.2** (Generalized Partition of Convex bodies [Louche and Ralaivola, 2015a]) *Let $\mathcal{C}$ be a closed convex body of dimension $D$ and $CG(\mathcal{C})$ its center of gravity. Let $h_{\boldsymbol{x}}$ a hyperplane of normal vector $\boldsymbol{x}$, $\|\boldsymbol{x}\| = 1$ and define the upper (resp. lower) partition $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) of $\mathcal{C}$ by $h_{\boldsymbol{x}}$ as*

$$\mathcal{C}^+ \doteq \mathcal{C} \cap \{\boldsymbol{w} \in \mathbb{R}^D : \langle \boldsymbol{x}; \boldsymbol{w} \rangle \geq 0\}$$
$$\mathcal{C}^- \doteq \mathcal{C} \cap \{\boldsymbol{w} \in \mathbb{R}^D : \langle \boldsymbol{x}; \boldsymbol{w} \rangle < 0\}.$$

*The following holds true: if $CG(\mathcal{C}) + \Lambda \boldsymbol{x} \in \mathcal{C}^+$ then*

$$\frac{Vol(\mathcal{C}^+)}{Vol(\mathcal{C})} \geq e^{-1}(1-\lambda)^D,$$

*where*

$$\Lambda = \lambda \Theta_D \frac{Vol(\mathcal{C}) H_{\mathcal{C}^+}}{R^D H_{\mathcal{C}^-}},$$

*with $\lambda \in \mathbb{R}$ an arbitrary real, $\Theta_D$ a constant depending only on $D$, $R$ the radius of the $(D-1)$-dimensional ball $\mathcal{B}$ of volume $Vol[\mathcal{B}] \doteq Vol\left[\mathcal{C} \cap \{\boldsymbol{w} \in \mathbb{R}^D : \langle \boldsymbol{x}; \boldsymbol{w} \rangle = 0\}\right]$ and $H_{\mathcal{C}^+} = \max_{v \in \mathcal{C}^+} \boldsymbol{v}^\top \boldsymbol{x}$ (resp. $H_{\mathcal{C}^-} = \min_{v \in \mathcal{C}^-} \boldsymbol{v}^\top \boldsymbol{x}$)*

*Proof.* The proof is a (non-trivial) extension of Grunbaum's one for Theorem 5.1 [Grunbaum, 1960] which can be found in full in appendix A.3 □

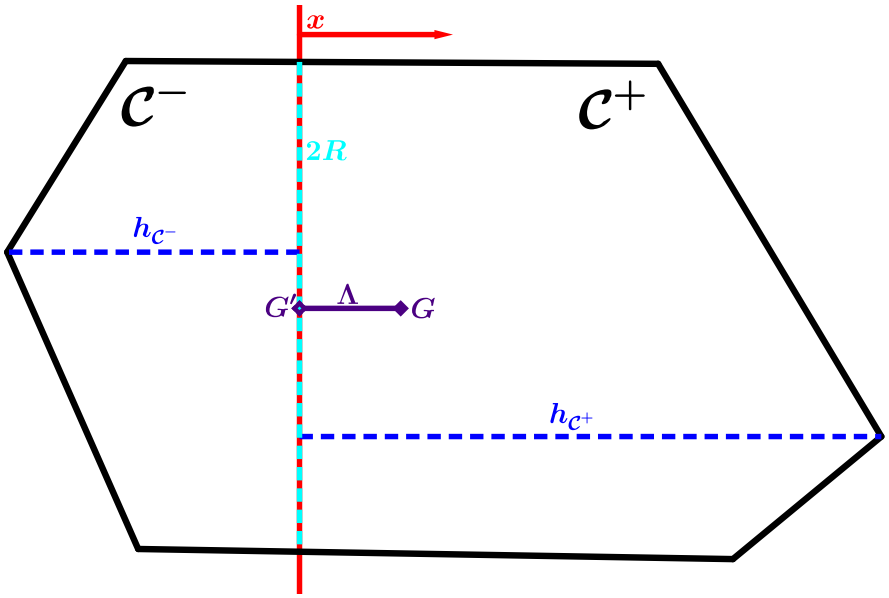



Figure 5.13 – *An schematic illustration of the different distances mentioned in Th. 5.2 on a two dimensional example. $G$ and $G'$ respectively represent the center of gravity and its approximation at a distance $\Lambda$. The red line represents the hyperplane of normal vector $x$ dividing $\mathcal{C}$ into $\mathcal{C}^+$ and $\mathcal{C}^-$. The blue distances represent $H_{\mathcal{C}^+}$ and $H_{\mathcal{C}^-}$, that is the distance between the red hyperplane and farthest point in $\mathcal{C}^+$ and $\mathcal{C}^-$. The light blue dashed line is the diameter of the 1-dimensional sphere of volume equal to the intersection between $\mathcal{C}$ and the red hyperplane; note that computing $R$ become a lot less straightforward as the dimensionality of $\mathcal{C}$ increase.*

We may add a few remarks to this result, especially regarding the $\Lambda = \lambda\Theta_D \frac{\text{Vol}(\mathcal{C})H_{\mathcal{C}^+}}{R^D H_{\mathcal{C}^-}}$ part. In a nutshell, $\Lambda$ is related to how the initial result of Th. 5.1 degrades with approximated centers of gravity. The terms $\Theta_D$ and $\text{Vol}(\mathcal{C})$ are directly related to $D$ and $\mathcal{C}$, in other words they do not depends on the approximate center of gravity. On the other hand the ratio $\frac{H_{\mathcal{C}^+}}{H_{\mathcal{C}^-}}$ and $R$ are more interesting. In substance, $H_{\mathcal{C}^+}$ (resp. $H_{\mathcal{C}^-}$) corresponds to the distance of the farthest point in $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) relatively to the hyperplane $h_x$ and we may immediately note that, contrary to Th. 5.1 the ratio between $\mathcal{C}^+$ and $\mathcal{C}^-$ is actually dependant of how $h_x$ intersects $\mathcal{C}$. The same goes for $R$ which is the radius of $(D-1)$-dimensional ball $\mathcal{B}$ of volume $\text{Vol}[\mathcal{B}] \doteq \text{Vol}\left[\mathcal{C} \cap \{w \in \mathbb{R}^D : \langle x; w\rangle = 0\}\right]$. A way to visualize this Ball is to consider the slice of $\mathcal{C}$ along $h_x$, that is the $D-1$ dimensional volume $\mathcal{C} \cap h_x$, $\mathcal{B}$ is a $(D-1)$-dimensional ball of same volume —for instance, if $\mathcal{C}$ is a 3-dimensional volume then $\mathcal{C} \cap h_x$ will describe a planar surface and $\mathcal{B}$ will be a disk whose area is the same than $\mathcal{C} \cap h_x$. More details on how $\mathcal{B}$ is constructed can be found in the proof of Th. 5.2 in the appendix or in the original works of [Grunbaum, 1960]. Moreover, another difference with corollary 5.1 lies in the fact that the partition ration is no longer independent of the dimension of $\mathcal{C}$ although this had to be expected because it was a convenient side effect of the result of theorem 5.1.

All in all, $\lambda$ is therefore intricately tied to the location of $h_x$ in $\mathcal{C}$ and, as a result, the intuition that an approximate center of gravity closer to its true location would imply better volume reduction property seems wrong.

The problem is actually more complex and depends on both the general shape of $\mathcal{C}$ and the orientation of $h_x$. For instance, for an elongated $\mathcal{C}$, $\lambda$ can vary a lot depending on whether $h_x$ intersects $\mathcal{C}$ along its long or short axis. From a practical perspective however we can reasonably assume that $\mathcal{C}$ has a shape regular enough for all possible values of $\lambda$ to be confined within a small interval, making the result of Th. 5.2 related to $\Lambda$ only —i.e. the distance between the true center of gravity and its approximation.

A relevant critic to the above discussion is to ask of the relevance of theorem 5.2 given that we advocate for an assumption similar to the smooth degradation of the result of theorem 5.1. We argue that theorem 5.2 provides insights on the limits of this assumption and identifies the important factors that may break it. As such, we gain a better understanding of the limits of our model and it is made easier to detect cases where model based on approximated center of gravity might under-perform.

## 5.4 Conclusion

This chapter stands apart from the rest of this thesis in the sense that it does not discuss Machine Learning directly. Instead, we focused on localization problem and Cutting Planes algorithm theory. We already have discussed in this chapter the strong links between localization and learning problems, and it should come without surprise that the next chapters build upon these similarities to develop new methods for machine learning. Additionally, we also discussed the matter of centroid of a convex body, something that is closely tied to the performance of Cutting Planes methods, and one of our contribution in this respect comes in the form of theorem 5.2 which is a non trivial and new extension of theorem 5.1 to approximated center of gravity. Although this result is motivated by linear classification considerations (see chapter 7 for the details of its use) we may underline that theorem 5.2 is not specific to machine learning and its relevance spans beyond the scope of this thesis and in particular to geometrical methods build around centers of gravity.

# Localization Methods applied to Machine learning

6

Contents



This chapter is a direct application of the ideas developed in Chapter 5 to the setting of machine learning. Namely, we will first discuss what interpretation we can give to the various notions introduced previously within the context and semantics of Machine Learning. We shall then unfold our discussion in two times. A first matter will be to discuss how centroids can be thought as classifier and consequently what underlying learning paradigms correspond to the Chebyshev and Gravity centers. A surprising result is that both correspond to known learning methods, with the Chebyshev's center being equivalent to a SVM classifier and the center of gravity being used as an approximation of the Bayes Point. The second point of this chapter comes from the observation that learning algorithms can be used to implement the query function of a Cutting Planes algorithm (Alg. 5) and as such it is possible to wrap classic learning methods into a Cutting Planes update Scheme. More precisely, we shall discuss this approach on two peculiar cases: SVM methods and the Perceptron algorithm. The former case yielded an already vastly discussed contribution [Joachims et al., 2009] that established remarkable improvement with respect to the classic SVM computational complexity; the latter case is a contribution of our own that has been published first in [Louche and Ralaivola, 2015a], its main interest is to show that Cutting

Planes are a natural way to enhance the Compression property of the perceptron algorithm without leaving the theoretical boundaries of theorem 1.3. The result is a new algorithm that is bounded in its total number of update by theorem 1.3 —consequently, this also bound the number of Cutting Planes queries— and exhibit a much more aggressive compression behaviour than a regular Perceptron.

## 6.1 Localization methods in the context of Machine Learning

### 6.1.1 Back to Machine Learning

As a preliminary to this chapter we shall make a proper transition from the matters discussed in the previous chapter —i.e. Cutting Planes algorithms and centroids— to linear classification. In many senses, this section mirrors Section 5.1.1 as we will unfold the localization setting we built up in chapter 5 within the context of binary linear classification.

#### Cutting Planes Algorithms for Learning

We shall revisit Cutting Planes algorithms in retrospect to the underlying learning problem we want to tackle. Let us first re-state the localization problem Cutting Planes algorithm are used for, that is to find a point $\boldsymbol{w}$ localized within a version space $\mathcal{W}$ defined with respect to a matrix $\mathbf{X} \in \mathbb{X}^N$ that can be thought as $N$ hyperplane's normal vectors (see also problems (5.1) (5.2) and (5.3)). On this topic we may underline that the matrix $\mathbf{X}$ is constructed from a learning problem's training set $\mathcal{S} \doteq \{(\boldsymbol{x}_i, t_i)\}_{i=1}^N$:

$$\mathbf{X} \doteq \begin{bmatrix} t_1 \boldsymbol{x}_1^\top \\ \vdots \\ t_N \boldsymbol{x}_N^\top \end{bmatrix}$$

Or, equivalently, from a training set $\mathcal{S}'$ in *normal form* (see definition 3.1) with data $t_i \boldsymbol{x}_i$ and labels $(+1)$ everywhere. Thus the core idea behind chapter 5 was to rest on this duality between points and hyperplanes to motivate the study of localization methods as potential learning algorithm. As such a conclusion of this discussion is that Cutting Planes algorithms do return a classifier $\boldsymbol{w}$ that is consistent with $\mathcal{S}$, yet we barely scratched the matter of how relevant the Cutting Planes properties are with respect to machine learning considerations.

The first point we shall discuss though, is the question of the specifics of the Cutting Planes oracle or, more precisely, how do we build a Cutting Planes oracle given our learning setting. To that question we may answer simply and remind that, by design, the Oracle function should be easy to compute and as such we go for the most straightforward solution, that is to directly search through $\mathbf{X}$ for a hyperplane that has a negative margin with the queried point. Equivalently, note that this amounts to finding a labelled example $(\boldsymbol{x}_i, t_i)$ in $\mathcal{S}$ on which the queried point —that is, a classifier— makes a prediction mistake. In anticipation of section 6.2.2 we may also mention two variations of this Oracle implementation that add a

minimal computational overhead. Namely, Cutting Planes Oracle that will consistently returns the cut with minimal or maximal (negative) margin when multiple cuts are possible. Intuitively, these two strategies lead to Cutting Planes Oracles that will either return the deepest cut possible or the least deep one and we shall discuss the relevance of these strategies in section 6.2.2.

### 6.1.2 On the property of Cutting Planes as Learning Algorithm

A matter we barely touched in the above discussion is that, similarly to the final output of a Cutting Plane algorithm, query points are akin to classifiers in a learning setting. More precisely, given the implementation of the oracle we propose, cuts are therefore data from $\mathcal{S}$ and consequently query points can be thought as classifiers that are consistent with a subset of $\mathcal{S}$. More precisely, from a learning perspective, the Cutting Planes update scheme can be summarized as follows: at each step, produce a classifier $w$ consistent with a partial dataset $\mathcal{S}_\mathcal{C}$ then ask for a new data from $\mathcal{S}$ on which $w$ errs. Where $\mathcal{S}_\mathcal{C}$ is the training set composed by the cuts returned by the Oracle so far, thus $\mathcal{S}_\mathcal{C} \subset \mathcal{S}$ is the learning counterpart of the search space $\mathcal{C}$ introduced in chapter 5.

Consequently, a defining feature of Cutting Planes methods when applied to learning problem is that they precisely proceed by building a simplified representation of $\mathcal{S}$ by selecting relevant data through Oracle queries. In particular, we argue that this behaviour gives rise to Sample Compression Scheme in the sense that the algorithm ultimately identifies a search space $\mathcal{C}$ that is a synthetic approximation of $\mathcal{W}$, that is $\mathcal{S}$, with an emphasis on making as few Oracle queries as possible.

**Property 6.1** (Cutting Planes Provides Samples Compression Scheme) *Any Cutting Plane algorithm such as depicted in algorithm 5 is a Sample Compression Scheme provided that the Oracle in line 7 and the Query Step in line 4 are deterministic.*

*Proof.* If the compression set is made of the training examples that define the cutting planes, this result is a direct consequence of the structure of Algorithm 5. A proof by induction that essentially hinges on the fact that, at each iteration $k$, the next query point is deterministically computed from $\mathcal{C}^k$ (only) gives the result. □

Additionally, the learning algorithm obtained with the assumptions of Proposition 6.1 is a *Process Sample Compression Scheme* [Littlestone and Warmuth, 1986], that is, even if we interrupt the learning before convergence has occurred, running the algorithm on the partial compression scheme obtained so far gives exactly the same predictor. Moreover, it is obviously an aim to have fast convergence of the localization procedure, where fast convergence means few iterations of the cutting-plane procedure. This directly translates into the idea of finding a point in the version space that is expressed as a combination of as few vectors as possible, which, by theorem 2.2, is very beneficial for generalization purposes. This is especially interesting in the context of centroids, and more specifically centers of gravity, that may come with guarantees on

the number of iterations (relatively to $\mathrm{Vol}(\mathcal{W})$), and therefore on $|\kappa(\mathcal{S})|$, to reach convergence.

Finally, we may state plainly a fact we only hinted at so far, that is finding a point in $\mathcal{C}$ is, per se, a learning task. Hence, any learning algorithm that operate the setting we have described so far —i.e. binary linear classification— is a potential implementation of the QUERY() function of algorithm 5. At this point in the discussion, a natural question that arises is the one of how usual classification methods may intertwine and benefit from theorem 2.2 when used as query strategies in a Cutting Planes setting and conversely what is the relevance of Cutting Planes specific query strategies with respect to the Machine Learning needs.

### 6.1.3 The case of CG and CC

Of the last section's concluding question, we shall answer the second part first. That is, in a binary linear classification setting, what is the underlying interpretation of centroids. To precise the setting, we may for a time forget about the overall Cutting Plane Scheme and focus on what happen on one iteration of the algorithm. More generally, because each iteration of a Cutting Planes algorithm amounts to solving a learning problem, we may consider the even more global setting where we have a training set $\mathcal{S}$ (independently of whether it is a sub-sample of a larger training set) and we are interested in the qualities of the centroids of $\mathcal{W}(\mathcal{S})$ from a learning point of view. In particular, the centroids we may consider are the Chebyshev's center and the center of gravity as they are the more relevant to our problem, as far as the Machine Learning literature goes.

#### The Chebyshev's center is a SVM classifier

We are going to argue in this section that the Chebyshev's center as we have defined it in section 5.3.2 is simply a rescaling of the classifier defined through the well known SVM formulation in section 1.3.2 equation (1.1).

Our argument rests on the geometrical interpretation of the margin in our setting. More precisely, we have defined the margin $\gamma_{\boldsymbol{x}}^{\boldsymbol{w}}$ between a point $\boldsymbol{x}$ and a classifier $\boldsymbol{w}$ as the value $\langle \boldsymbol{x}; \boldsymbol{w} \rangle$ (see definition 1.8) or alternatively, if we assume $\boldsymbol{w}$ to be of unit norm ($\|\boldsymbol{w}\|_2 = 1$), the margin is the distance between a point $\boldsymbol{x}$ and the hyperplane of normal vector $\boldsymbol{w}$. In an equivalent way, we can consider the dual interpretation we have discussed so far where data are akin to hyperplanes and classifier to points. Under this new perspective, the converse also hold true, and assuming a data $\boldsymbol{x}$ of unit norm ($\|\boldsymbol{x}\|_2^2$) the margin $\gamma_{\boldsymbol{x}}^{\boldsymbol{w}}$ is the distance of $\boldsymbol{w}$ from the hyperplane defined by $\boldsymbol{x}$. Because in our setting the classification rule of a linear classifier only depends on the sign of $\langle \boldsymbol{x}; \boldsymbol{w} \rangle$ we can freely rescale all datapoint to have a unit norm and therefore we will be reasoning over $\gamma_{\boldsymbol{x}}^{\boldsymbol{w}}$ as the distance between a point $\boldsymbol{w}$ and the hyperplane of normal vector $\boldsymbol{x}$. Therefore the SVM solution, that is a classifier with maximal margin over $\mathcal{S}$, seems to intuitively match the idea of largest inscribed sphere.

**Theorem 6.1** (Chebyshev's center and SVM) *For any training set $\mathcal{S}$ and associated version space $\mathcal{W}(\mathcal{S})$, where $\mathcal{S}$ and $\mathcal{W}$ defined as usual (see respectively section 1.2.2 and 5.1.1),*

*let denote $\boldsymbol{w}_{SVM}$ the (hard-margin) SVM's solution on $\mathcal{S}$ as defined in the canonical SVM formulation in section 1.3.2 and $CC(\mathcal{W})$ the Chebyshev's center of $\mathcal{W}$ (see definition 5.3). Then the following hold true:*

$$CC(\mathcal{W}) = \frac{\boldsymbol{w}_{SVM}}{1 + \|\boldsymbol{w}_{SVM}\|_2}$$

*Proof.* The idea of the proof is to show that for a fixed norm, the center of the ball of largest margin over $\mathcal{S}$ is always a rescaling of $\boldsymbol{w}_{\text{SVM}}$.

Let $\alpha \doteq \|\boldsymbol{w}_{\text{SVM}}\|_2$, the first step is to show that there is no vectors $\boldsymbol{w}'$ of norm $\|\boldsymbol{w}'\|_2 = \alpha$ such that $\gamma_{\mathcal{S}}^{w'} > \gamma_{\mathcal{S}}^{w}$ (see definition 1.8).

Formally, assuming the converse to be true, then there is a real $\epsilon > 0$ such that $\gamma_{\mathcal{S}}^{w'} = \gamma_{\mathcal{S}}^{w_{\text{SVM}}} + \epsilon$. Alternatively, we have that for all $\boldsymbol{x}$ in $\mathcal{S}$, $\langle \boldsymbol{w}'; \boldsymbol{x} \rangle \geq \langle \boldsymbol{w}_{\text{SVM}}; \boldsymbol{x} \rangle + \epsilon \geq 1 + \epsilon$, thus there exists a vector $\widetilde{\boldsymbol{w}}' \doteq \boldsymbol{w}'/(1+\epsilon)$ of norm $\|\widetilde{\boldsymbol{w}}'\|_2 < \alpha$ such that, for all $\boldsymbol{x}$ in $\mathcal{S}$

$$\begin{aligned}
\langle \widetilde{\boldsymbol{w}}'; \boldsymbol{x} \rangle &= \left\langle \frac{1}{1+\epsilon} \boldsymbol{w}'; \boldsymbol{x} \right\rangle \\
&= \frac{1}{1+\epsilon} \langle \boldsymbol{w}'; \boldsymbol{x} \rangle \\
&\geq 1
\end{aligned}$$

which is a contradiction with the definition of $\boldsymbol{w}_{\text{SVM}}$.

Similarly, for any real $\beta = \lambda\alpha$ with $\lambda > 0$, we can show that the point of fixed norm $\beta$ of maximal margin over $\mathcal{S}$ is $\lambda \boldsymbol{w}_{\text{SVM}}$. Again, assuming the converse to be true would imply the existence of $\boldsymbol{w}'$ as defined above by a simple matter of rescaling.

Additionally, let $\gamma_{\mathcal{S}}^{CC(\mathcal{W})}$ the margin realized by $CC(\mathcal{W})$ on $\mathcal{S}$. Let write $\beta$ the norm of $CC(\mathcal{W})$ and by definition of $\mathcal{B}_1$ we know that $CC(\mathcal{W})$ is at a distance $1 - \beta$ from the edge of $\mathcal{B}_1$. From the previous arguments, we know that the point of maximum margin with norm $\beta$ is $\beta \boldsymbol{w}_{\text{SVM}}/\alpha$, that is to say, $CC(\mathcal{W}) = \beta \boldsymbol{w}_{\text{SVM}}/\alpha$. Moreover we have by calculation that for all $\boldsymbol{x}$ in $\mathcal{S}$, $\langle \beta \boldsymbol{w}_{\text{SVM}}/\alpha; \boldsymbol{x} \rangle \geq \beta/\alpha$ or, put otherwise $\gamma_{\mathcal{S}}^{CC(\mathcal{W})} \geq \beta/\alpha$.

Finally, from the definition of the Chebyshev's Center, finding the value of $\beta$ is a matter of maximising at the same time the margin $\gamma_{\mathcal{S}}^{CC(\mathcal{W})}$ of $CC(\mathcal{W})$ and its distance to $\mathcal{B}_1$:

$$\beta = \arg\max_{\beta > 0} \min(\beta/\alpha, 1 - \beta)$$

where the first (resp. second) term is linearly increasing (resp. decreasing) with $\beta$, thus the solution hold for

$$\begin{aligned}
\frac{\beta}{\alpha} &= 1 - \beta \\
\Leftrightarrow \beta &= \frac{\alpha}{1+\alpha}
\end{aligned}$$

and putting everything back together yield that

$$\begin{aligned}
CC(\mathcal{W}) &= \frac{\beta}{\alpha} \boldsymbol{w}_{\text{SVM}} \\
&= \frac{1}{1+\alpha} \boldsymbol{w}_{\text{SVM}} \\
&= \frac{\boldsymbol{w}_{\text{SVM}}}{1 + \|\boldsymbol{w}_{\text{SVM}}\|_2}
\end{aligned}$$

□

We shall note that this equivalence is a well known and discussed result in the literature [Tong and Koller, 2001, Herbrich et al., 2001, Rujan, 1997]; however we did not found, to our greater surprise, any formal proof of the above theorem where the rescaling term is made explicit, hence our choice to present the full proof here.

### 6.1.4 Bayes Point Machine and Center of Gravity

Contrary to the Chebyshev's center, centers of gravity have been studied in machine learning directly for their learning properties, albeit recently. Works like [Rujan, 1997, Herbrich et al., 2001, Minka, 2001a, Brinker, 2004] were instrumental in the late 90's and early 2000's in enabling geometric approaches to machine learning and these works are a notable inspiration of the studies presented here. Considering the version space's center of gravity for learning purpose can be traced back to Bayesian models and more precisely to the idea of *Bayes Classification Strategy* which is defined as the optimal classification strategy, in the sense that, it corresponds to the strategy that predict the most likely label according to a vote across all the hypotheses consistent with the data, that is the version space.

**Definition 6.1** (Bayes Classification Strategy) *Given a fixed loss $l$, a training set $\mathcal{S}$ and the corresponding version space $\mathcal{W}$, and assuming*[1] *that $\mathbb{H}$ is the class of linear classifier with uniform prior distribution $\mathbb{P}_{\mathbb{H}}$ and posterior distribution $\mathbb{P}_{\mathbb{H}|\mathcal{S}}$ such as*

$$\mathbb{P}_{\mathbb{H}|\mathcal{S}}(h_{\boldsymbol{w}}) = \begin{cases} \frac{1}{Vol(\mathcal{W})} & if \quad \boldsymbol{w} \in \mathcal{W} \\ 0 & if \quad \boldsymbol{w} \notin \mathcal{W} \end{cases}$$

*The Bayes Classification Strategy (BCS) maps any point $\boldsymbol{x} \in \mathbb{X}$ to either $(+1)$ or $(-1)$ as follow:*

$$\begin{aligned} h_{BCS}(\boldsymbol{x}) &\doteq \arg\min_{y \in \mathbb{Y}} \mathbb{E}_{h_{\boldsymbol{w}} \overset{i.i.d}{\sim} \mathbb{P}_{\mathbb{H}|\mathcal{S}}} \left[ l(sign(h_{\boldsymbol{w}}(\boldsymbol{x})), y) \right] \\ &\doteq \arg\min_{y \in \mathbb{Y}} \int_{\boldsymbol{w} \in \mathcal{W}} l(sign(\langle \boldsymbol{w}; \boldsymbol{x} \rangle), y) d\boldsymbol{w} \end{aligned}$$

It is known that the Bayes Classification Strategy provably outperforms all other classification strategy [Herbrich et al., 2001] although it usually does not correspond to an element of $\mathbb{H}$ and as such, it is not a valid solution to our problem. More importantly, it also means that deciding for a label requires to go through all classifiers in $\mathcal{W}$ for each new data, which is computationally expensive [Graepel et al., 1999]. In [Herbrich et al., 2001] the authors propose a solution to this problem by projecting the Bayes Classification Strategy onto the space $\mathbb{H}$ of linear classifiers. More precisely, they define the *Bayes Point* as the element of $\mathbb{H}$ that mimics the best the Bayes Classification Strategy over the entire data distribution $\mathscr{D}_t$, that is the classifier $\boldsymbol{w}$ that minimizes the loss $\lambda$ with respect to the labels

[1] In itself, Bayesian theory does not require these assumptions to hold and this definition is usually given in a slightly more general form. However, in order to streamline the discussion, we focus on our specific problem rather than the general theory (see [Herbrich et al., 2001, Mitchell, 1982] for more details).

computed from the Bayes Classification Rules as it is defined in definition 6.1

**Definition 6.2** (Bayes Point) *Given $\mathcal{S}$, $\mathcal{W}$, $\mathbb{H}$ and $\mathbb{P}_{\mathbb{H}|\mathcal{S}}$ defined as before and $\mathscr{D}_t$ the underlying data distribution over $\mathbb{X}$. We define the* Bayes Point *as the linear classifier $h_{\boldsymbol{w}^{BP}}$ of normal vector $\boldsymbol{w}^{BP}$ such that:*

$$\boldsymbol{w}^{BP} \doteq \arg\min_{\boldsymbol{w}\in\mathbb{X}} \mathbb{E}_{\boldsymbol{x}\overset{i.i.d}{\sim}\mathscr{D}} \left[ \mathbb{E}_{h_{\boldsymbol{w}'}\overset{i.i.d}{\sim}\mathbb{P}_{\mathbb{H}|\mathcal{S}}} \left[ l(sign(h_{\boldsymbol{w}'}(\boldsymbol{x})), sign(h_{\boldsymbol{w}}(\boldsymbol{x}))) \right] \right]$$

Alternatively, the authors of [Rujan, 1997, Rujan and Marchand, 1999] provide a geometrical interpretation of both the Bayes Classification Strategy and the Bayes Point that help the understanding of these notions. Consider the version space $\mathcal{W}$ and a new datapoint $\boldsymbol{x}$ to label; if the hyperplane of normal vector $\boldsymbol{x}$ does not cut through $\mathcal{W}$ the problem is trivial and all classifiers in $\mathcal{W}$ predict the same label for $\boldsymbol{x}$, otherwise the Bayes Classification Rules advocates to split $\mathcal{W}$ in two set $\mathcal{W}_+$ and $\mathcal{W}_-$ depending on the sign of $\langle \boldsymbol{w}; \boldsymbol{x} \rangle$ and $h_{\text{BCS}}$ then predicts $(+)$ if $\text{Vol}(\mathcal{W}_+) \geq \text{Vol}(\mathcal{W}_-)$ and $(-)$ else. Additionally the authors introduce the notion of *Bayes-line*, that is any hyperplane of normal vector $\boldsymbol{x}$ that splits $\mathcal{W}$ in two equal volumes $\text{Vol}(\mathcal{W}_+) = \text{Vol}(\mathcal{W}_-)$. In a nutshell, Bayes-lines correspond to the limit cases where the Bayes Classification Strategy goes from predicting $(+)$ to $(-)$ and intuitively any classifier $\boldsymbol{w} \in \mathcal{W}$ that aims to mimic the Bayes Classification Strategy must lie on those Bayes-lines, in other words if $\boldsymbol{x}$ defines a Bayes line we want a classifier $\boldsymbol{w}$ such that $\langle \boldsymbol{w}; \boldsymbol{x} \rangle = 0$. Obviously, there is absolutely no guarantee that all the Bayes-lines intersect in one single point; if such situation arises, it means that the Bayes Classification Strategy actually belong to $\mathcal{W}$ and can be reduced to a single classifier $\boldsymbol{w}$ which is in most cases not possible. The Bayes Point defined above is an attempt to find a point that is the closest of all Bayes-Lines intersections; where closest is relative to the loss $l$ used (see also Fig 6.1). Notably, the notion of Bayes Line is dependent of the underlying distribution $\mathscr{D}$ and as such computing the Bayes Point require the capacity to massively sample over $\mathscr{D}_t$ which is something not possible in the general case.

In practice, the authors of [Watkin, 1993, Herbrich et al., 2001] note that if $\mathscr{D}_t$ is Gaussian[2], then the center of mass is a very good approximation of the Bayes Point. Moreover, the geometric argument of [Rujan, 1997] on Bayes Line echoes the volume property of the center of gravity (theorem 5.1), yet for any form of result to hold in this respect, we have to assume the Bayes lines to be uniformly distributed over all directions. In other words, the center of gravity is a valid approximation for any distribution that ensures a uniform distribution of the normalized data $\boldsymbol{x}/\|\boldsymbol{x}\|$ over the unit sphere, which is obviously the case when $\mathscr{D}_t$ is Gaussian.

Bayes Point Machines —BPM for short— are defined as the family of algorithms that build upon these ideas and return as classifier the center of gravity of the version spaces [Herbrich et al., 2001]. Experimental results in [Herbrich et al., 2001] tends to demonstrate the good generalizations properties of BPM, and [Rujan, 1997, Rujan and Marchand, 1999] give empirical evidence of the relevance of the billiard algorithm and the

[2]By Gaussian, we mean that the points $t(\boldsymbol{x})\boldsymbol{x}$ are distributed over $\mathbb{X}$ according to a Gaussian distribution

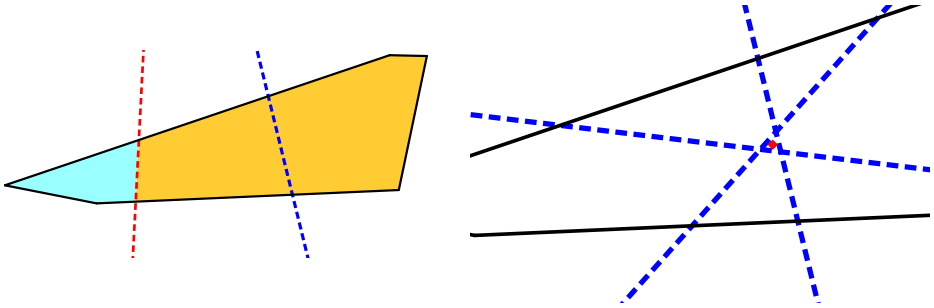


Figure 6.1 – *A depiction of the Bayes Classification Strategy. The left panel represents a version space of classifier and the red and blue lines two data in this representation. The red line split the version space in two parts corresponding to whether the classifiers in the version space would make a positive prediction (orange area) or a negative one (blue area). The Bayes Classification Rule acts as a majority vote over the version space and would assign a positive label to the red data. The blue line however is a* Bayes Line *and splits the version space in two equal part, thus the result of the majority vote is indecisive. The right figure depicts the same version space (focused on the area of interest, for convenience) with three Bayes Lines that do not intersect on a unique point. In this case, no linear classifier can perfectly emulate the Bayes Classification Rule. The red dot represents the Bayes Point, a linear approximation of the Bayes Classification Rule.*

closeness (in terms of geometrical distance) between the Bayes Point, the Center of Gravity, and the Billiard's approximation of the Center of Gravity. Finally, we shall also mention [Minka, 2001a, Minka, 2001b] which introduce the *Expectation Propagation* algorithm, a method that encompasses, among other things, billiard-like estimation of the Center of Gravity.

On this front, we shall note that the idea of restricting $\mathcal{W}_0$ to the unit ball $\mathcal{B}_1$ proposed in section 5.3.3 —as opposed to [Herbrich et al., 2001, Rujan, 1997] where the billiard is reset when the ball escape $\mathcal{B}_1$— is also advocated by [Minka, 2001b] in which the author observes enhanced stability with this approach.

## 6.2 Cutting Planes Powered Machine Learning

In the previous section, we have discussed the relevance of Cutting Planes methods and notions in the context of Machine Learning. In particular, we have been interested in how centroids were more or less knowingly used as classifier in the past years. This is yet another hint at the fact that Machine Learning problems could be efficiently tackled as localization problems, and that the relevant literature holds great theoretical and practical solution for our field. This section aims to take the reasoning a step further and we shall now focus on how to fully integrate a Cutting Planes methods into learning problems. Namely, we will discuss the matter of performing learning directly with Cutting Plane algorithms and how the iterative approach of Cutting planes methods can be an advantage when combined with classical learning approaches.

### 6.2.1 Cutting Plane and SVM

We will first discuss a well established result from [Joachims et al., 2009] built upon a smart interfacing between a Cutting Planes algorithm and a SVM.

This section essentially reviews the works of [Joachims and Yu, 2009] and [Joachims et al., 2009] for the case of binary classification. Whether our readers are interested in more details or how the results presented here may port to other settings, we refer them to the aforementioned papers. Notably, the setting presented here is a bit different than the one established so far, nonetheless, the resulting algorithm can be thought as a Cutting Planes algorithms, where the queried point is $CC(\mathcal{W})$, however the Oracle is a bit different than what we discussed before.

First, let us introduce a variation of the SVM formulation we have introduced in section 1.3.2 that is known under the name of *soft margin SVM*. In a nutshell, we want to allow for errors to be made by introducing a *slack* variable to the objective functional. This slack variable will account for error in the training set by adding a penalty to the objective functional we minimize. Nonetheless, this will allow for erroneous solution given that the penalty paid for errors does not outweighs the gain for a better overall solution where said penalty is controlled by a hyper-parameter $C$. Ultimately, this setting ensure that every problem has a feasible solution, even when the data are not linearly separable.

**Definition 6.3** (Soft-margin SVM formulation) *Given a training set $\mathcal{S} \doteq \{(\boldsymbol{x}_i, t_i)\}_{i=1}^N$ and a predetermined scalar value $C$, we define $\boldsymbol{w}_{\text{soft-SVM}}$ and $\xi_1, \dots, \xi_N$ as the solution to the following optimization problem:*

$$\begin{aligned} & \min_{\boldsymbol{w}, \xi_i \geq 0} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{N}\sum_{i=1}^N \xi_i \\ s.t. \quad & \forall i \in [N] : t_i \langle \boldsymbol{w}; \boldsymbol{x}_i \rangle \geq 1 - \xi_i \end{aligned}$$

Computationally speaking, this is also a quadratic problem that can be solved with any quadratic programming solver although we have introduced $N$ additional slack variables in the form of $\xi_1, \dots, \xi_N$. Solving the problem as it is would imply having a resolution time that scale polynomially in the number of variable, hence in $N$. On the other hand, for each data point corresponds a different constraint, thus the number of constraints is linear in $N$. Remark that, Cutting Planes can easily cope with a large number of constraints (see section 5.2 and the discussion on Sample Compression Scheme in section 2.2), however the increased number of variables is detrimental to the overall efficacy of the algorithm. The key idea of [Joachims et al., 2009] is to rewrite the soft-margin SVM problem with exponentially more constraints, but fewer variables: the so-called *structural* SVM formulation

**Definition 6.4** (Structural SVM formulation) *Given a training set $\mathcal{S} \doteq \{(\boldsymbol{x}_i, t_i)\}_{i=1}^N$ and a predetermined scalar value $C$, we define the structural SVM problem as follow:*

$$\begin{aligned} & \min_{\boldsymbol{w}, \xi \geq 0} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\xi \\ s.t. \quad & \forall \boldsymbol{c} \in \{0,1\}^N : \frac{1}{N}\sum_{i=1}^N \boldsymbol{c}_i t_i \langle \boldsymbol{w}; \boldsymbol{x}_i \rangle \geq \frac{1}{N}\sum_{i=1}^N \boldsymbol{c}_i - \xi \end{aligned}$$

Moreover Joachims and Yu show that the solution $(\boldsymbol{w}, \xi)$ of a Structural SVM problem is actually also a solution to the soft-margin SVM one, where $\xi = \sum_{i=1}^{N} \xi_i$ ([Joachims and Yu, 2009], Theorem 1).

Ultimately, the Cutting Plane algorithm will rely on an oracle that is capable of generating constraints on the go, this will notably avoid the otherwise prohibitive computational cost of evaluating all of the $2^N$ constraints at each step. Namely, for a given $\boldsymbol{w}$ the oracle build the vector $\boldsymbol{c} \in \mathbb{R}^N$ as follow:

$$\boldsymbol{c}_i \doteq \begin{cases} 1 & \text{if} \quad t_i \langle \boldsymbol{w}; \boldsymbol{x}_i \rangle < 1 \\ 0 & \text{otherwise} \end{cases}$$

and return the corresponding constraint. We will not give a full rewriting of the algorithm here as it involves a slightly different setting than ours, especially in regard to the soft-margin SVM and would require to spend a substantial amount of time on alternate definition and notation for something that is auxiliary to our main point. The message of this section however is that, if allowed to return an approximate solution $(\boldsymbol{w}, \xi)$ that achieves an objective value (definition 6.4) within an $\epsilon$ range of optimal one, then the number of Cutting Planes iterations required is constant with respect to $N$ and polynomial with other factors. Experimental results hint at an overall execution time roughly linear in $N$ though, which underlines that both theoretically and empirically the embedding of learning algorithm into Cutting Planes update scheme can be computationally interesting. More importantly, this work constitutes a first positive result and a strong motivation to unravel this thread in research, especially given that, to much of our surprise, we could not find any work similar to [Joachims et al., 2009] for BPM. It is our belief that similar results could be achieved with, in addition, far stronger theoretical results due to the natural properties of centers of gravity. Namely, convergence should comes naturally without the need of an approximate solution. Moreover, adding a Cutting-Plane selection strategy on top of the regular BPM algorithm could help alleviate the usually high cost of the billiard sampling method. Moreover, our setting is flexible enough to allow for error tolerant formulation akin to the soft-margin SVM and it is an avenue for improvement to look a this specific problem for BPM when used in conjunction with Cutting-Planes methods similarly to [Joachims et al., 2009].

### 6.2.2 Perceptron and Cutting Planes

#### Motivations

Although the BPM way is an immediate and interesting extension of the previous section, we preferred to focus ourselves on another founding algorithm of machine learning, the Perceptron algorithm. Our reasons for this are multiple and we shall share some in order to help motivate our approach. Looking back at the previous section, we may notice that despite its sound geometrical interpretation, SVM methods do not have strong complementary properties with Cutting Planes algorithms. Not only the SVM solution lacks the theoretical guarantees of a center of gravity, but it is known to already embed property such as Compression Scheme. On the other hand, Perceptrons would fully benefit from this compression prop-

erty and at the same time their online update scheme naturally fits the query scheme of Cutting Planes. In particular, our approach rests on the idea that once a Perceptron has reached convergence, it may be restarted from its latest iteration if new data are made available which is in sharp contrast to SVM methods that typically require to go through the entire learning procedure a second time.

### The Algorithm

The solution we propose is to use a tweaked perceptron algorithm as the QUERY() function, with a Cutting Planes Oracle following the implementation discussed in section 5.2 as depicted in algorithm 6.

**Algorithm 6** Top : A Perceptron-based localization algorithm. Bottom : The slightly modified Perceptron algorithm for compression scheme.

**Require:** $\mathcal{S}$ a training set.
**Ensure:** Find $w \in \mathcal{W}$ (see section 5.1.1)
$\mathcal{C}^0 \leftarrow \mathcal{B}_1$
$k \leftarrow 0$, $\boldsymbol{w}^0 \leftarrow \mathbf{0}$
**repeat**
  $\tilde{\boldsymbol{w}}^k \leftarrow \text{PERCEPTRON}(\tilde{\boldsymbol{w}}^{k-1}, (\boldsymbol{x}_{n_0}, t_{n_0}), \cdots, (\boldsymbol{x}_{n_k}, t_{n_k}))$ ▷ Query point
  $\boldsymbol{w}^k \leftarrow \tilde{\boldsymbol{w}}^k / \|\tilde{\boldsymbol{w}}^k\|$
  **if** $\boldsymbol{w}^k \notin \mathcal{W}$ **then**
    Pick a cutting plane $\boldsymbol{x}_{n_k}$ s.t. $t_{n_k} \langle \boldsymbol{w}^k; \boldsymbol{x}_{n_k} \rangle < 0$ ▷ Oracle
    $\mathcal{C}^{k+1} \leftarrow \mathcal{C}^k \cap \{\boldsymbol{w} : t_{n_t} \langle \boldsymbol{w}; \boldsymbol{x}_{n_k} \rangle \geq 0\}$
    $k \leftarrow k + 1$
  **end if**
**until** $\boldsymbol{w}^k \in \mathcal{W}$
**return** $\boldsymbol{w}^k$

**function** PERCEPTRON($\boldsymbol{w}^{\text{start}}, (\boldsymbol{x}_{n_0}, t_{n_0}), \cdots, (\boldsymbol{x}_{n_k}, t_{n_k})$)
  $k \leftarrow 0$
  $\boldsymbol{w}^0 \leftarrow \boldsymbol{w}^{\text{start}}$
  **while** $\exists n_i : t_{n_i} \langle \boldsymbol{w}^k; \boldsymbol{x}_{n_i} \rangle < 0$ **do**
    $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + t_{n_i} \boldsymbol{x}_{n_i}$
    $k \leftarrow k + 1$
  **end while**
  **return** $\boldsymbol{w}^k$
**end function**

The aforementioned tweaks come into play to ensure that the Cutting Planes query scheme and the Perceptron update process intertwine correctly. Namely, this algorithmic compound can be seen in two different ways: on one hand it is a Cutting Planes algorithm that run a (tweaked) Perceptron to select its next query point, on the other hand it is a Perceptron algorithm that is fed by a Cutting Planes oracle. In order to do this, we have to take full advantage of the online property of the Perceptron.

For the sake of the discussion, let say that we have some black-box that can produce a stream of data and that this black-box is used to fed a Perceptron algorithm. Initially, the black-box produces only a handful,

of distinct data and simply loops on these points, hence the Perceptron will be fed with duplicates until convergence —note that convergence will happen though, even with duplicates (see theorem 1.3 and 6.2). Once all points are correctly classified, the black-box will add a new point to its pattern until convergence is, again, achieved. This is akin to an overly supervised execution of the Perceptron, where a new data is introduced only when all the previous ones are correctly classified. Of course, the new data could cause updates that may induce mistakes on the rest of the dataset in such a way that one does not know how many iterations it will take to reach once again a solution each time a new data is introduced. Practically speaking, the black-box do the exact same thing than the Cutting Plane scheme of algorithm 6: at each step $k$ it feds the Perceptron on a loop with the same data until a solution is achieved then, query a new data from the oracle, restart the Perceptron from its previous solution and start over. We illustrate this by rewriting algorithm 6 from a Perceptron centric perspective in algorithm 7. See also the flowchart presented in Figure 6.2.

**Algorithm 7** The same algorithm (see algorithm 6) from a Perceptron's perspective. The set $\mathcal{E}$ corresponds to the search space in the sense that $\mathcal{C} = \mathcal{B}_1 \cap \left[\bigcap_{x_i \in \mathcal{E}} \{\boldsymbol{w} : \langle \boldsymbol{w}; \boldsymbol{x}_1 \rangle \geq 0\}\right]$

**Require:** $\mathcal{S}$ a training set.
**Ensure:** Find $w \in \mathcal{W}$ (see section 5.1.1)

$k \leftarrow 0$, $\boldsymbol{w}^0 \leftarrow 0$
**while** $\exists \boldsymbol{x}_i \in \mathcal{E} : t_i \langle \boldsymbol{w}^k; \boldsymbol{x}_i \rangle < 0$ **do**
  $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + t_i \boldsymbol{x}_i$ ▷ Perceptron update
  $k \leftarrow k + 1$
  **if** $\forall \boldsymbol{x}_i \in \mathcal{E} : t_i \langle \boldsymbol{w}^k; \boldsymbol{x}_i \rangle \geq 0$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \text{CPOracle}(\mathcal{S}, \boldsymbol{w}^k, \mathcal{E})$ ▷ Ask for a new datapoint
  **end if**
**end while**
**return** $\boldsymbol{w}^k / \|\boldsymbol{w}^k\|_2$

**function** CPOracle($\mathcal{S}$, $\boldsymbol{w}$)
  **if** $\exists \boldsymbol{x}_i \in \mathcal{S} : t_i \langle \boldsymbol{w}; \boldsymbol{x}_i \rangle < 0$ **then** ▷ equiv. $\boldsymbol{w}/\|\boldsymbol{w}\|_2 \notin \mathcal{W}$
    **return** $\boldsymbol{x}_i$
  **else**
    **return** $\emptyset$
  **end if**
**end function**

### Convergence

A noteworthy feature of the algorithm we propose is its convergence speed. Intuitively, it might seem that we are left with a very inefficient algorithm that requires to solve a Perceptron problem after each Cutting Planes iteration; in addition, the lack of volume reduction guarantees for querying the Oracle with the Perceptron classifier also seems to discard any convergence result on the number of Cutting Planes iteration.

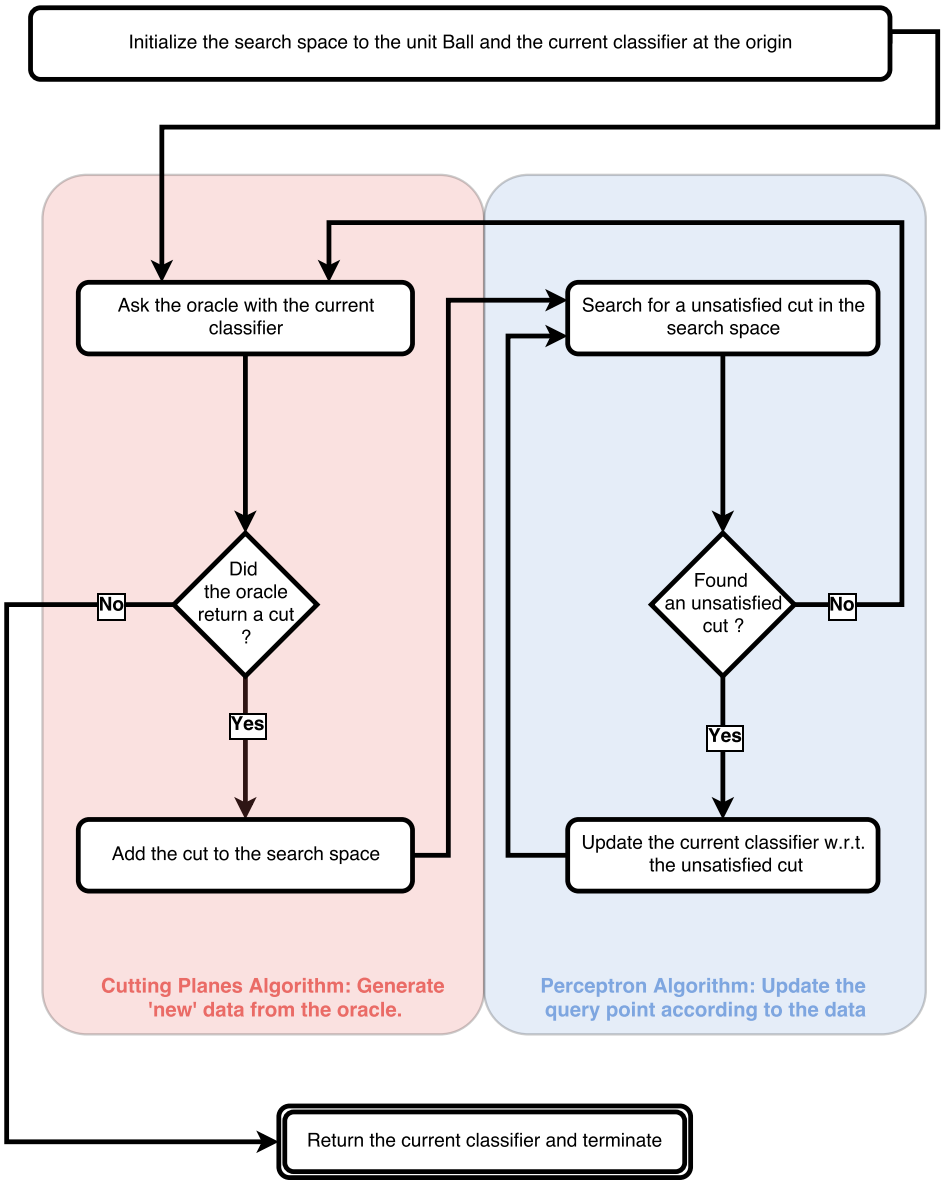



Figure 6.2 – *A synthetic depiction of how the Perceptron and Cutting Planes algorithms interact.*

In reality, we can dismiss both of these matters as both the number of Perceptron's iterations and Oracle queries are theoretically bounded in the algorithm we propose. We have argued before (sections 5.2) that the intermediate localization problems are just a means to drive the resolution of the global problem. In particular, we leverage the online property of the Perceptron algorithm to build upon this idea by restarting each new execution from the last queried point. Remarkably, theorem 1.3 allows for a theoretical bound to hold on the total number of Perceptron updates required to solve the problem; that is, the number of time lines 18 in algorithm 6 is executed across the whole algorithm, from the beginning to the output of the final classifier.

**Theorem 6.2** (Mistake Bound) *Let $\mathcal{S} \doteq \{(\boldsymbol{x}_i, t_i\}_{i=1}^N$ a training set such that there exists a classifier $\boldsymbol{w}^*$, $\|\boldsymbol{w}^*\| = 1$ and a scalar $\gamma_{\mathcal{S}}$ such that $\forall (\boldsymbol{x}_i, t_i) \in \mathcal{S} : t_i \langle \boldsymbol{w}^*; \boldsymbol{x}_i \rangle > \gamma_{\mathcal{S}}$ —alternatively, the largest inscribed sphere in $\mathcal{W}_0$ of center $\boldsymbol{w}^*$, $\|\boldsymbol{w}^*\| = 0$ has a radius of at least $\gamma_{\mathcal{S}}$.*

*Define $M$ the number of Perceptron updates performed by the Perceptron-based Localization Algorithm 6 (i.e. $M$ is the number of times line 18 of* Perceptron *of Algorithm 6 is executed). Then the following holds:*

$$M \leq \frac{R^2}{\gamma_{\mathcal{S}}^2}$$

*with $R \doteq \max_{\boldsymbol{x}_i \in \mathcal{S}} \|\boldsymbol{x}_i\|$.*

*Proof.* Remind that theorem 1.3 [Block, 1962, Novikoff, 1962] holds for any sequence of points independently of its size or generation process; in particular, the result holds when the sequence is made of duplicated and dependent data. Consider the rewriting of the algorithm given in Alg. 7 and more precisely the sequence point used for update in line 3. Let us write this sequence $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_M$. By construction, this sequence is not independently distributed and does contain duplicates, however it is nonetheless sampled from $\mathcal{S}$, hence there exists a solution $\boldsymbol{w}^*$ of norm 1 that achieves a margin of $\gamma_{\mathcal{S}}$ on this sequence. The theorem is then obtained from the fact that algorithm 7 acts as a regular Perceptron with respect to this peculiar sequence of points. □

Hence, on any learning problem, our algorithm enjoys the same mistake bound than a vanilla Perceptron and, given that the overhead from Oracle queries is usually negligible, each update is as fast as a regular Perceptron update. Nonetheless, it also has the advantages of Cutting-Planes methods, that is, it is an efficient Sample Compression Scheme. From a higher picture, theorem 6.2 shows that our algorithm is a practical implementation of the idea that the intermediate localization problems should drive the resolution of the problem toward an efficient solution.

### Experimental Results

Here, we present some empirical simulations based on the modified perceptron algorithm we introduced above (algorithm 6).

We generate a toy dataset of $1,000$ 2-dimensional datapoints. Each point is uniformly drawn on a 20-by-20 square centered at the origin. We

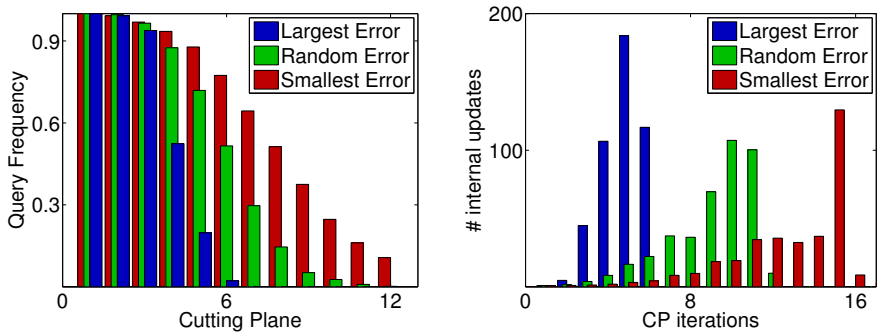



Figure 6.3 – Left *: for each value i the bar represents the empirical probability (over* 1,000 *runs) to query at least i Cutting Planes.* Right*: each bar represents the number of internal Perceptron updates computed after each Cutting Planes loop.*

label this dataset according to a classifier $\boldsymbol{w}^*$ uniformly drawn over the unit circle. In order to have only positive labels, negative examples are reflected through the origin. We then enforce a minimal margin $\gamma$ by pruning examples $\boldsymbol{x}_i$ for which $\langle \boldsymbol{w}^*; \boldsymbol{x}_i \rangle < \gamma$. This last modification allows us to have some control over the size of the version space $\mathcal{W}$. The downside of this is that we no longer have exactly 1,000 datapoints (though during our experiments we noted that the size of the dataset stays mostly the same for reasonable margin values).

For these experiments, we use the Perceptron-based Localization algorithm (Algorithm 6). We implement it with three different oracle strategies for selecting cutting planes. The first strategy (which we call *Largest Error*) picks the cutting plane with the lowest margin. The second one (*Smallest Error*) picks the Cutting Planes with the highest negative margin, that is to say points that are incorrectly classified but close to the decision boundary. Finally, the third one (*Random Error*) simply picks a Cutting Planes with negative margin at random. It should also be noted that our instantiation of the Perceptron algorithm picks the update vector that realizes the lowest margin for its internal update —line (18) of Perceptron in algorithm 6. This is mostly an arbitrary choice and we only mention it for the sake of reproducibility.

The first experiment consists in a single run over a dataset of margin $\gamma = 0.1$. We monitor both the number of Cutting Planes generated and the number of internal Perceptron updates for each cutting plane. The presented results are averaged over 1,000 runs.

The left pane of Figure 6.3 supports the soundness of our approach in the case of a compression scheme with no more than 6 Cutting Planes for the best strategy (Largest Error). Additionally, we can observe a sharp decrease after the third Cutting Planes with this strategy and 80% of the time, only 4 Cutting Planes are required to model the dataset. In contrast, the right-hand side of Figure 6.4 reveals a trade-off between the number of Cutting Planes used and the number of internal updates for each Cutting Planes. We observe a smooth shift across our three strategies with Smallest Error putting the emphasis on small number of internal updates. In all respect, the Random Error strategy acts as a middle ground between the two other extreme approaches.

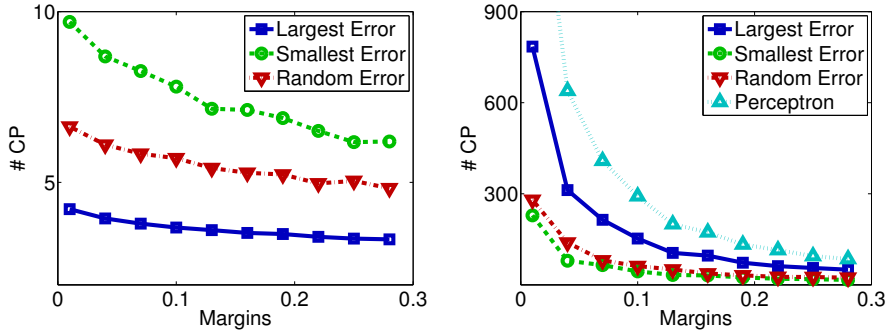



Figure 6.4 – Left: *The average number of Cutting Planes used for each strategy with respect to the value of* $\gamma$. Right: *the total number of internal updates with respect to* $\gamma$. *The fourth plot corresponds to a regular Perceptron.*

For the second experiment the margin[3] is variable with values between 0.01 and 0.3. We also monitor the total number of internal updates rather than the *per Cutting Planes* value for the three strategies and a regular Perceptron Algorithm [4]. Remind that these four plots are bounded from theorem 6.2, however the plots are far below their common theoretical bound and plotting it here would not have been possible given the scale.

The previously observed behavioral shift across the three strategies is confirmed by figure 6.4. Additionally, some relative robustness is observed with respect to $\gamma$, especially when the emphasis is put on querying a small number of Cutting Planes. It is interesting to note that the Random Strategy makes nearly as few updates as Smallest error while still querying a —relatively— low number of Cutting Planes. Finally, all three strategies are making slightly less updates than the regular Perceptron.

## 6.3 Conclusion

The take home message of this chapter is simple and can be summarized as follow: computational geometry literature is relevant to machine learning theory. Through this chapter, this is an idea we have illustrated in two different ways. First, we showed how centroids have a relevance to machine learning and how reasoning over the geometric properties of the version space may yield efficient classifiers. Moreover, the connection between Chebyshev's center and SVM is noteworthy and we shall underline how both machine learning and computational geometry developed similar solutions to equivalent problems. Secondly, we discussed that wrapping classic learning methods within a Cutting Planes update scheme can be beneficial from a learning point of view. Notably we discussed how Cutting Planes can bring improved convergence speed when interfaced with SVM methods [Joachims and Yu, 2009], or provide Sample Compression Scheme when interfaced with the Perceptron Algorithm [Louche and Ralaivola, 2015a]. It is our belief that we have just scratched the top of the possible interconnections between those two fields and it is

[3] See [Tong and Koller, 2001] for a discussion on how the margin may relate in practice the Vol($\mathcal{W}$)

[4] More precisely, we use the exact same Perceptron than the one used for the internal loop but ran on the full dataset

an interesting research project in itself to further expand on these ideas. A tangible first step in this direction is through Analytic Center Cutting Planes Methods which have recently been considered for their learning properties [Fanzi et al., 2009].

# Cutting-Plane Powered Active Learning

7

Contents



This chapter is devoted to a slightly different setting than the ones explored in the previous chapters, namely we will be interested here in the setting of Active Learning. In a continuation of the previous chapter, we argue that in addition to what we already have discussed, Cutting Planes methods are also very close to Active Learning procedures. More precisely, both rely on a query mechanic given some Oracle and seek to iteratively improve their current solution. We will show that Active Learning can mostly be stated in terms on Cutting Planes and the interfacing between the two settings lies in the specifics of the Oracles. Nonetheless, one of our result is to show that a viable Cutting Planes Oracle can be built from an Active learning Oracle and as such Cutting Planes methods can be used to solve Active Learning problems. From this approach stems a new family of Active Learning algorithm based on geometrical considerations that was first proposed in [Louche and Ralaivola, 2015a]; notably one specific instance of this new family is an adaptation of the BPM algorithm to the setting of Active Learning that was independently discussed in [Brinker, 2004]. Additionally, our algorithmic contribution comes with a novel theoretical analysis based on Cutting Planes theory that asserts of the soundness of our endeavour as well as helps in explaining the results previously observed in [Tong and Koller, 2001]'s SIMPLE algorithm. In

addition, we shall note that our approach borrows from both the Expected Model Change and Uncertainty Sampling theories of active learning and aims to achieve an acceptable trade-of between the theoretical guarantees of the former and the computational simplicity of the latter.

## 7.1 Active Learning: Motivations and General Framework

*Supervised* learning —that is, the setting we have discussed until now— carries on the implicit assumptions that: 1) data and labels are obtained at the same time, through the same process 2) there is no interaction between the learning algorithm and the data/label source. While the reasons for such assumptions are multiple, they are arguably too restrictive and unnatural. In particular, data and labels come usually from two different sources and the former may be some order of magnitude cheaper to obtain than the latter. This is typically the case when data are produced by measurement devices but requires human expertise to interpret, a textbook example of this is the case of medical imaging where collecting data is cheap and can be performed by low-wage employees but establishing a diagnosis requires specialized physician knowledge.

Relaxing only the first assumption leads to the settings of *Unsupervised* and *Semi-Supervised* learning that are well-studied and vivid topics in Machine Learning, but fall outside of the context of this thesis (see [Zhu, 2005] for an in-depth discussion of the these topics). On the other hand, relaxing both assumptions is known as *Active Learning*.

More precisely, in the *Active Learning* setting we assume that, contrary to labels, data are marginally cheap to obtain. The idea is to provide the algorithm with unlabeled data at first and then let the algorithm decide which labels it needs; in other words, the learning algorithm has to *actively* query the labels. Sample complexity is then measured in term of the number of label queries. The underlying idea being that allowing the algorithm to choose the relevant labels —as opposed to generating an already labelled dataset— should allows for better sample complexity.

A way to put this into perspective is to think of Supervised Learning as a wasteful process where many points in $\mathcal{S}$ are redundant from a learning point of view. Active Learning is built upon the idea that by querying the labels of carefully picked datapoint only, one can avoid this data redundancy without hindering its learning capabilities.

The usual observation about *Active Learning* is that, in most cases, the sample complexity is exponentially reduced comparatively to *Supervised Learning* [Settles, 2010].

## 7.2 Active Learning Strategies

### 7.2.1 Two Strategies of interest

*Active Learning* is a vastly studied topic, and results for this setting in the recent years are numerous. Exposing here the totality of recent research on the topic would be both cumbersome and unpractical. Hence, giving an

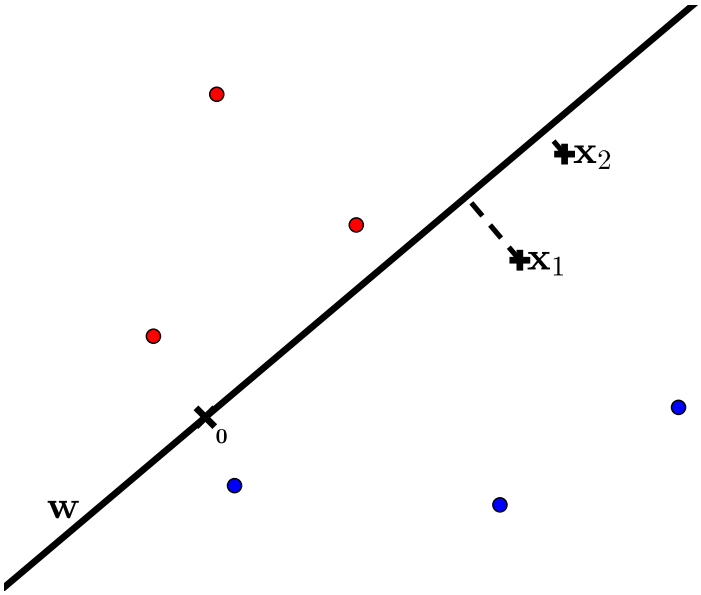



Figure 7.1 – *An illustration of the uncertainty query strategy. In this example,* $\mathbf{0}$ *is the origin, blue and red dots correspond to positive and negative examples;* $\boldsymbol{x}_2$ *and* $\boldsymbol{x}_1$ *are two unlabeled datapoints. In this case, an uncertainty based strategy would query* $\boldsymbol{x}_2$ *because the point is closer to the current hypothesis (*$\boldsymbol{w}$*), thus its label is considered more uncertain.*

extensive review of the field here is unrealistic and the interested reader can refers to the survey of [Settles, 2010] for a more complete review.

Instead, we shall focus on two particular *Active Learning Strategies* that are closely related to the solution we will propose. Namely, we will be interested in the so-called *Uncertainty Sampling* [Lewis and Gale, 1994] and *Expected Model Change* [Settles et al., 2007] approaches.

Formally, in *Active Learning* one has access to a training set $\mathcal{S}_0 \overset{\text{i.i.d}}{\sim} \mathscr{D}^N$ —alternatively, $\mathcal{S}_0 \doteq \{\boldsymbol{x}_i\}_{i=1}^N$. In other words, $\mathcal{S}_0$ is a training set without labels. In order to cope with $\mathcal{S}_0$ the learner has access to a *Labelling Oracle* $O : \mathbb{X} \to \mathbb{Y}$ such that $O(\boldsymbol{x}) \doteq t(\boldsymbol{x})$. Thus, the problem of *Active Learning* is to devise a query strategy for the *Oracle* that minimizes the number of queries.

In *Uncertainty Sampling* the queries are focused on datapoints for which the label is uncertain. Typically, this approach works well with probabilistic settings, where classifiers not only output the class but also a confidence estimation for the prediction. The idea is to proceed in rounds, where each round consists in a prediction step where $\mathcal{S}_0$ is labeled by the current classifier, a query step where the *oracle* is queried on the most uncertain pair label/datapoint, and a learning step where the classifier is re-trained using only the queried points and labels. We also refers the readers to works like [Settles, 2010, Lewis and Gale, 1994].

On the other hand, the *Expected Model Change* operates differently and aims at finding a query point that would guarantee the largest model change, independently of the Oracle's label. In other world, what we are truly interested in is to make the most informative query model-wise and while the idea is theoretically sound, such strategy is in general computationally costly as it implies to compute what the new model would be for every point/label combinations.

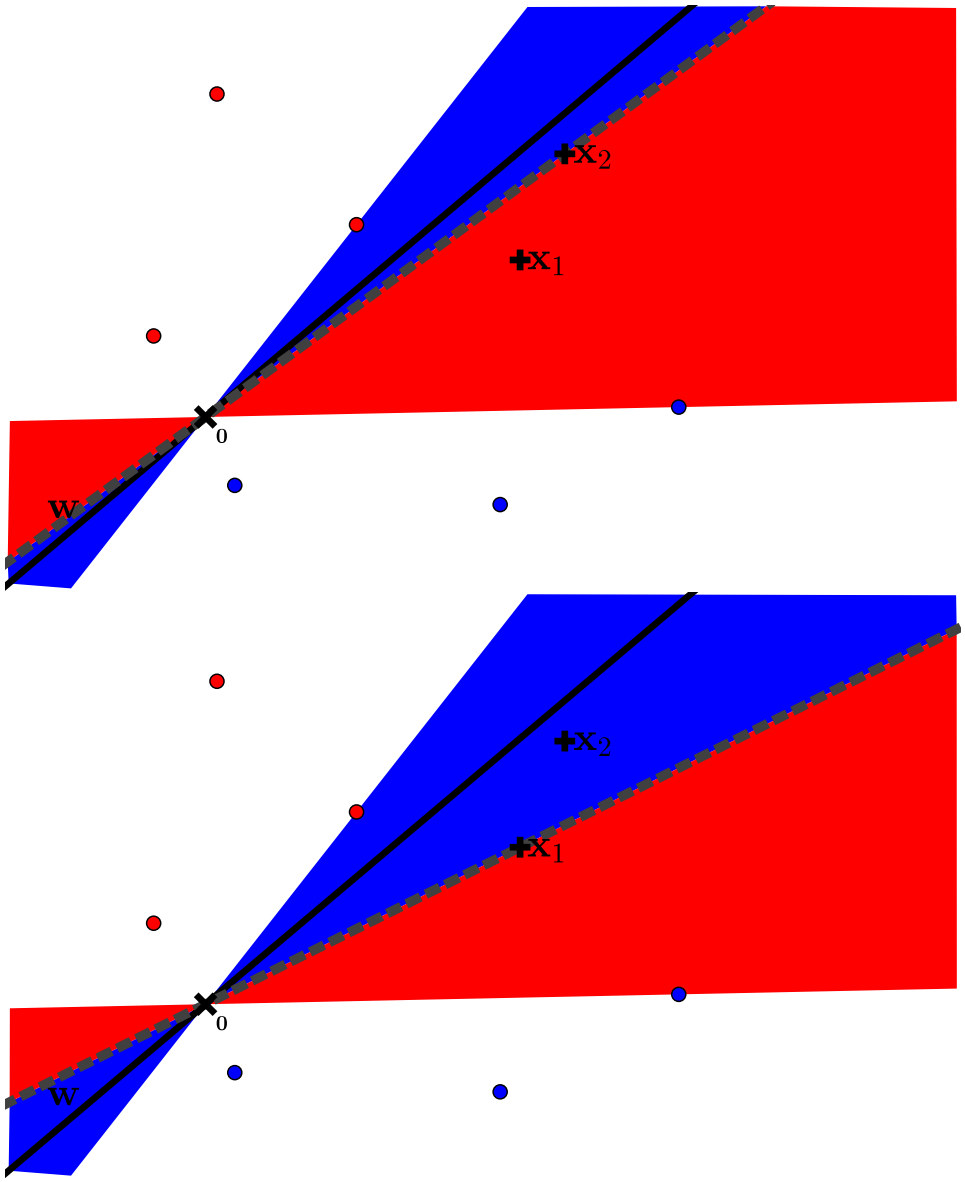



Figure 7.2 – *An illustration of the Expected Model Change Strategy. This is the same example than Fig. 7.1. The red and blue areas represent the set of consistent solutions after receiving a positive (red area) or negative (blue area) for $\boldsymbol{x}_2$ (top) or $x_1$ (bottom). In this example querying $\boldsymbol{x}_1$ would halve the space of consistent solutions independently of the label received —i.e. the blue and red areas on the bottom have the same size— this is not the case for $\boldsymbol{x}_2$. Hence, an Expected Model Change Strategy would query $\boldsymbol{x}_1$ because doing so would guarantee a substantial reduction in the number of possible classifiers, whereas in the case of $\boldsymbol{x}_2$ it would depend of the label returned by the Oracle.*

Notably, both of these strategies have been used with linear classifiers. In the case of *Uncertainty Sampling*, the margin —that is, the distance to the hyperplane defined by $\boldsymbol{w}$— is acting as a confidence measure while, in the case of *Expected Model Change* one usually keeps track of the variations of the *Version Space*'s volume $\mathrm{Vol}(\mathcal{W})$ as a way to measure changes in the model. More precisely, informative queries are precisely the ones that invalidate as many classifiers as possible in $\mathcal{W}$. A general update scheme for both strategies in the bi-class linear settings is given in algorithm 8 and 9; where the expectation's distribution over $\{+1, -1\}$ in algorithm 9, line 6 is usually arbitrary and specific to the implementation.

**Algorithm 8** A General Uncertainty Sampling Query Scheme for Linear Classifier

**Require:** An unlabeled training set $\mathcal{S}_{\mathbb{X}} \doteq \{\boldsymbol{x}_i\}_{i=1}^{N}$, an oracle $O$ and a learning procedure LEARN.

1: $\mathcal{S} \leftarrow \emptyset$
2: $\boldsymbol{w} \leftarrow \mathbf{0}$
3: **loop**
4: $\quad \boldsymbol{x} \leftarrow \arg\min_{\boldsymbol{x} \in \mathcal{S}_{\mathbb{X}}, \boldsymbol{x} \notin \mathcal{S}} |\langle \boldsymbol{w}; \boldsymbol{x} \rangle|$ ▷ Select the most uncertain point
5: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{(\boldsymbol{x}, O(\boldsymbol{x}))\}$ ▷ Query the label
6: $\quad \boldsymbol{w} \leftarrow \textsc{learn}(\mathcal{S})$ ▷ Update the classifier
7: **end loop**
8: **return** $w$

**Algorithm 9** A General Expected Model Change Query Scheme for Linear Classifier

**Require:** An unlabeled training set $\mathcal{S}_{\mathbb{X}} \doteq \{\boldsymbol{x}_i\}_{i=1}^{N}$, an oracle $O$, a learning procedure LEARN and some difference measure $(\cdot \Delta \cdot)$ over subsets of $\mathbb{H}$.

1: $\mathcal{S} \leftarrow \emptyset$
2: $\mathcal{W} \leftarrow \mathbb{H}$
3: **loop**
4: $\quad$ Update $\mathcal{W}$ according to:

$$\mathcal{W} \doteq \{h \in \mathbb{H} : \forall (\boldsymbol{x}, t(\boldsymbol{x})) \in \mathcal{S}, h(\boldsymbol{x}) = t(\boldsymbol{x})\} \tag{7.1}$$

5: $\quad$ For each $\boldsymbol{x} \in \mathcal{S}_{\mathbb{X}}$ compute:

$$\mathcal{W}_{\boldsymbol{x}}^{+} \doteq \mathcal{W} \cap \{h \in \mathbb{H} : h(\boldsymbol{x}) = +1\}$$
$$\mathcal{W}_{\boldsymbol{x}}^{-} \doteq \mathcal{W} \cap \{h \in \mathbb{H} : h(\boldsymbol{x}) = -1\}$$

6: $\quad \boldsymbol{x} \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{S}_{\mathbb{X}}} \left[\mathbb{E}_{i \in \{+,-\}} \left[\mathcal{W} \Delta \mathcal{W}_{\boldsymbol{x}}^{i} \mid \boldsymbol{x}\right]\right]$
7: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{(\boldsymbol{x}, O(\boldsymbol{x}))\}$ ▷ Query the label
8: **end loop**
9: $\boldsymbol{w} \leftarrow \textsc{learn}(\mathcal{S})$
10: **return** $w$

Recently, Tong and Koller [Tong and Koller, 2001] have proposed to implement both of these strategies with SVM methods. Their implementation of *Uncertainty Sampling* yielded one of the most successful modern

*Active Learning* algorithm; sometimes referred as the SIMPLE *algorithm*. Part of the success of SIMPLE is its overall high performances coupled with a relatively low computational cost of *Uncertainty Sampling*. However, they also observed that SIMPLE notably fails in some cases and proposed a more robust solution to this problem based on the idea of *Expected Model Change*. Interestingly, their solution replaces the expectation part of the algorithm (Alg. 9, line 6) by an equilibrium criterion over the sizes of $\mathcal{W}_{\boldsymbol{x}}^-$ and $\mathcal{W}_{\boldsymbol{x}}^+$. Particularly, they propose two different criteria, thus defining two other methods that, despite being more robust, are also far more computationally intensive.

Finally, note that the work of Tong and Koller is deeply tied to our own approach and we will come back to it in greater detail later. Especially a question that is central to this problem is the one of how these two strategies, and by extension the different methods they propose, relate to each others.

### 7.2.2 A Version Space Approach to Active Learning

As discussed above, the so-called *Expected Model Change Query Scheme* is build upon the idea that a good query should have a huge impact on the model learned. In other word, in order to decide which data will be queried, one compute each possible Oracle's response ($+1$ or $-1$ in our case) and measure how this impacts the resulting predictive model once this new information has been incorporated.

Speaking of model in the case of linear bi-class problem is akin to speaking of *version space*, that is, the set of all linear classifier that are consistent with the labelled data. The point of this section is to lay down the basis for re-interpreting Active Learning problems as Cutting Planes ones. Namely, we will argue that Active Learning problems can be reduced to the problem of shrinking the version space as effectively as possible.

We already have discussed how localization and learning problems are related (Section 5.1.1). The bottom-line of this discussion is that labelled datapoint can be seen as hyperplane's normal vectors (that is $t_i \boldsymbol{x}_i$) and the problem of learning is reduced to the problem of localizing a point to the positive side of every hyperplane at once; the (convex) set of all these points being the version Space $\mathcal{W}$.

Given this setting, we shall distinguish the information carried by a label from the information carried by a datapoint. Namely, a datapoint $\boldsymbol{x}$ is seen as normal vectors and thus informs on the general direction the corresponding hyperplane whereas its label acts as a switch: if it is $+1$ the hyperplane has normal vector $\boldsymbol{x}$, if it is $-1$ the hyperplane is reversed and has normal vector $-\boldsymbol{x}$. In other words, depending on the label, the positive and negative sides of the hyperplane get switched, without changing its orientation. Moreover, we will henceforth denote that a datapoint is of unknown label by writing $\pm\boldsymbol{x}$ and $h_{\pm\boldsymbol{x}}$ will refer to the non oriented hyperplane —that is, without positive or negative side— of normal vector $\boldsymbol{x}$ or $-\boldsymbol{x}$ equivalently.

Practically speaking, querying a new point amounts to add a new hyperplane to the problem. If this hyperplane does not intersect the version space, the query is non-informative: it does not change the learned model

at all. Conversely, the query is informative if it intersects the version space. Note that the distinction between informative and non-informative queries does not require the labels to be known in advance; because every classifier in the version space predicts the same label for those queries, we can already infer their label. Intuitively, we can measure how informative a given query is by quantifying how much of the version space is discarded once the label is known. More formally, for a version space $\mathcal{W}$, a given query will split it into two subsets $\mathcal{W}_1$ and $\mathcal{W}_2$ such that $\mathcal{W}_1 \cup \mathcal{W}_2 \doteq \mathcal{W}$ — note that one of those may be the empty set, in the case of non-informative queries. Which set is discarded and which is not is then determined by the label returned by the Oracle. The volume of the discarded set is directly related to how helpful the query is and the smaller the version space, the lower the uncertainty about the objective classifier. Ultimately, when the version space is small enough so that no potential query intersects it, the problem is solved.

Strategies that fall under the Expected Model Change paradigm are built upon these version space considerations. Because the goal is to query as few labels as possible, they revolve around computing the expected reduction of the version space, given all possible queries and labels. Practically speaking, it means looking for a query which will lead to a sizable volume reduction of the version space independently of the label returned by the Oracle. In particular, we want to avoid queries that unevenly split the version space as their relevance is essentially tied to whether the oracle return the convenient label.

Although Expected Model Change Strategies are theoretically and experimentally sound, they very computationally demanding; at each query step, the algorithm has to learn a model for every possible point/label pair in order to determine the best data to ask oracle with. Practically though, some heuristic can be used in order to speed up the process, yet the general procedure remains the same and relies on solving a learning problem for every point/label couple considered.

Our idea takes its ground in the similarities between Cutting Planes methods and active learning. Both follow the same scheme of query/update with respect to some Oracle. Moreover both rely onto the fast reduction of some solution space for their efficiency. In the following, we will show that Cutting Planes theory can be leveraged in order to devise a strategy that both fall under the Expected Model Change setting and does not require solving multiple learning problem at each step.

## 7.3 Active Learning And Cutting Planes

### 7.3.1 A state of Active Learning Methods, and how they relate to CP

Amidst the the prolific works related to Active learning (see, e.g. [Settles, 2012] for a review) we may single out a few contributions our work draw inspiration from; they share the common feature of focusing on/exploiting the geometry of the version space. The query strategies proposed by [Golovin and Krause, 2010] and [Gonen et al., 2013] are based on multiple estimations of the volume of the (potential) version space,

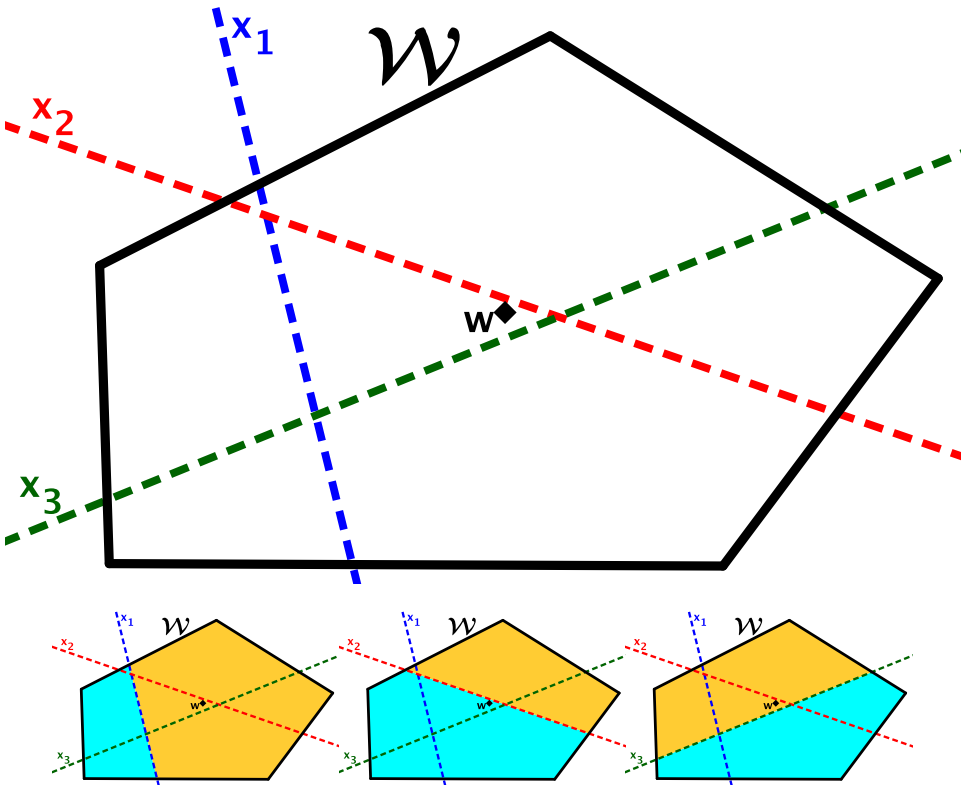



Figure 7.3 – *A version space $\mathcal{W}$ with the hyperplanes corresponding to three data $\boldsymbol{x}_1$ (blue), $\boldsymbol{x}_2$ (red) and $\boldsymbol{x}_3$ (green). The black dot represent the current classifier $\boldsymbol{w}$. The expected model change paradigm requires to evaluate the size of the cyan and orange areas for each data (the three bottom figures) and then query the data corresponding to the most balanced split, that is $\boldsymbol{x}_3$ in the present case. On the other hand the uncertainty query paradigm queries the label of the closest data of $\boldsymbol{w}$, that is $\boldsymbol{x}_2$ in this case. Our method seeks to achieve the best of those two approaches and relies on learning a classifier $\boldsymbol{w}$ such that the closest data is very likely to be the one yielding the most even split of $\mathcal{W}$.*

which, as discussed above might be computationally expensive when added together.

In comparison, in the active learning strategy we derive from the general Cutting Planes approach, we compute our queries from an approximated center of gravity of the version space, which is computationally equivalent to a single volume estimation. On the other hand, the work of [Balcan et al., 2007] that proposes a margin-based query strategy derived from uncertainty sampling theory (see [Lewis and Gale, 1994]) also provides theoretical justifications of such strategies and gives insights on the foundations the work of [Tong and Koller, 2001] hinges on.

In particular, our work is closely related to the one of [Tong and Koller, 2001] which pioneered the route to make Active learning procedure from Cutting Planes algorithm, even though the connection with Cutting Planes methods was not clearly identified in this work. Notably, the same idea has been successfully used with Cutting Planes for active Boosted Learning [Trapeznikov et al., 2011]. We may also mention [Brinker, 2004] which tackles the problem of active learning in a similar way —the connection with Cutting Planes is never made explicit though— and share some of our conclusion on the general idea on Active Learning methods based on Centers of gravity.

A by-product of our active learning procedure is that we now solve a Bayes Point Machine (BPM) problem [Herbrich et al., 2001] at each step $k$ by finding the center of gravity of the current version space $\mathcal{W}^k$. Therefore, we can turn our active learning procedure into a full Active Learning algorithm —that we dub ACTIVE-BPM— straightforwardly by using the center of gravity for classification.

Note that this is one of many possible instantiations of our procedure, which is nonetheless of interest as it is the BPM-counterpart of the ACTIVE-SVM algorithm of [Tong and Koller, 2001]. Moreover, theorem 5.2 provides a general guideline to systematically query the training point that comes with the best volume reduction guarantees. This is a theoretically sound and viable strategy for Active Learning that comes with a theoretical bound on the induced volume reduction, the lack of which was an essential limit of the Chebyshev's center-based method of [Tong and Koller, 2001]. Hence, our contribution is to show how the Cutting Planes literature and its accompanying worst-case convergence analyzes may give rise to theoretically supported query strategies that do not have to hinge on margin-based arguments.

### 7.3.2 Tuning the Cutting Planes for active learning

The solution we propose rests on the idea that both the Active Learning query scheme and Cutting Planes methods are based on the same general process. Namely, both can be divided in two parts:

- A query phase, where some Oracle provides insightful information relatively to a specific request
- An update phase, where the model process this newly acquired information.

The process then repeats itself over and over until a definitive solution is found. This section addresses the small differences that exists between Active Learning Query Scheme and Cutting Planes methods in a way that will allow for Cutting Planes results to be applied to Active Learning problems.

More precisely, we may identify that Active Learning and Cutting Planes settings differ on the specifics of the Oracle. In the Cutting Planess setting, one can ask the Oracle about virtually any point. The Oracle then responds with a separating hyperplane such that the queried point lie on one side of the hyperplane, and the (true) version space on the other. On the other hand Active Learning does not allow for arbitrary queries: only the points that are present in the dataset are allowed to be queried; moreover, from a version space perspective, those datapoints are akin to hyperplanes and the oracle feedback is much more limited in the sense that it only contains information on which side of the hyperplane is the positive one. In other words, given a hyperplane $h_{\pm \boldsymbol{x}}$ with normal vector of unknown sign $\pm \boldsymbol{x}$, the oracle simply lift the sign uncertainty.

The following of this section focuses on establishing that Cutting Planes methods are efficient at solving Active Learning problems. Our analysis is based on a modified Cutting Planes Oracle that is both easy to implement from an Active Learning Oracle and non-detrimental to the efficacy of Cutting Planes methods. That peculiar Oracle therefore acts as an interface between the two settings and ultimately allows for Active Learning problems to be solved through Cutting Planes methods.

Consider a slightly modified Cutting Plane Oracle $O'$ that differs from a regular Cutting Plane Oracle in the following way: it has access to a list of available cuts through their normal vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ but the list is twisted and some cuts are actually reversed, hence if the Oracle chooses to return a cut $h_{\boldsymbol{x}_i}$, it may actually end up returning a cut $h_{-\boldsymbol{x}_i}$. In order to compensate for that flaw, $O'$ is designed in such a way that it always returns the cut that is the closest to the queried point $\boldsymbol{w}$.

Building an Oracle that operate in the same way than $O'$ from an Active Learning Oracle is easy. Let $\mathcal{S}_0$ be an unlabeled dataset $\mathcal{S}_0 \doteq \{\boldsymbol{x}_i\}_{i=1}^N$ and $\tilde{O}_t$ the Active Learning oracle such that

$$\tilde{O}_t(\boldsymbol{x}_i) = t(\boldsymbol{x}_i)$$

. We define $O'$ as

$$O'(\boldsymbol{w}) = \tilde{O}_t \left( \min_{\boldsymbol{x} \in \mathcal{S}} \|\boldsymbol{x} - \boldsymbol{w}\| \right) \min_{\boldsymbol{x} \in \mathcal{S}} \|\boldsymbol{x} - \boldsymbol{w}\|$$

In other words, when queried with $\boldsymbol{w}$, $O'$ look for the datapoint $\boldsymbol{x} \in \mathcal{S}$ that is the closest of $\boldsymbol{w}$, query $\tilde{O}_t$ on this point and return the hyperplane of normal vector $t_i \boldsymbol{x}_i$.

We claim that $O'$ is sufficient for Cutting Planes algorithm to work correctly if, at a given step $k$, the algorithm query $\boldsymbol{w}^k \doteq CG(\mathcal{C}^k)$, the center of gravity of $\mathcal{C}^k$; moreover, the algorithm will likewise enjoy a reduction of $\mathcal{C}$'s volume that is exponential in the number of steps taken. Where the search space $\mathcal{C}^k$ in the context of Active Learning correspond to the current version space $\mathcal{W}^k$. We back our claim by establishing that, at each

step the volume of $\mathcal{C}$ is nearly halved in most cases, regardless of whether $O'$ return a twisted cut.

First consider the case where, for a given query $\boldsymbol{w}$, $O'$ returns a neutral cut $h_{\pm\boldsymbol{x}}$: clearly this case does not depends on whether $h_{\pm\boldsymbol{x}}$ is twisted or not because $\boldsymbol{w}$ lies right in between the two sides of the hyperplane; in other words, the side (positive or negative) on which $\boldsymbol{w}$ is located doest not matter and theorem 5.1 (see also [Grunbaum, 1960]) holds as it stands for this case. Alternatively, when the Oracle does not return a neutral cut, that is when $h_{\pm\boldsymbol{x}}$ is a deep cut, we propose to consider a hypothetical query point $\boldsymbol{w}'$ for which $h_{\pm\boldsymbol{x}}$ is a neutral cut. Because $O'$ always return the closest cut to $\boldsymbol{w}$ we know that $\boldsymbol{w}'$ will be as close as possible to $\boldsymbol{w}$ and our idea is to deal with $\boldsymbol{w}'$ as if it was an approximation of $\boldsymbol{w}$. From theorem 5.2 we can extend the result of theorem 5.1 to approximate centers of gravity such as $\boldsymbol{w}'$ and obtain guaranteed volume reduction of $\mathcal{W}$.

What theorem 5.2 means in this context is that independently of the cut $h_{t_i\boldsymbol{x}_i}$ returned by $O'$ we can always take a point $\boldsymbol{w}'$ such that $\boldsymbol{w}' = CG(\mathcal{W}) + \lambda t_i \boldsymbol{x}_i$ and apply theorem 5.2 to $\boldsymbol{w}'$. A problem subsists however: we have previously discussed how theorem 5.2 does not guarantee that the distance between the true center of gravity and its approximation does not forcefully relates to the quality (volume-wise) of the approximation and in order to account for that, $O'$ shouldn't return the closest cut to $CG(\mathcal{W})$ but rather the best one according to the value of $\Lambda$ in theorem 5.2. Faced with this problem we can go two ways: either modify $O'$ such that it queries the best datapoint according to theorem 5.2 or adopt a practical perspective and consider that, in most cases, $H_{\mathcal{C}^+}/H_{\mathcal{C}^-}$ and $R$ are seemingly constant when considering all the possible cuts in $\mathcal{S}$. The former approach usually require to solve some computationally expensive problems even though smart upper bounding may result in relative precise and computationally efficient solution. On the other hand, empirical results (Section 7.4 and [Brinker, 2004]) tend to show that the latter solution is reasonable on real Active Learning problems and that is the one we will adopt for the present matter.

Putting everything together, we can write down a new Active Learning algorithm based upon Cutting Planes methods (Alg. 10) that provably reduces the size of $\mathcal{W}^k$ with each query by at least a constant factor. Our algorithm relies on computing a center of gravity at each step and for the sake of clarity we use the center of gravity as a classifier in the predictive model, hence the name of ACTIVE-BPM. Nonetheless, any learning procedure can be adapted our query scheme thus spanning a whole new family of Active Learning algorithms based on Cutting Planes method. We shall also mention that our algorithm mirror [Tong and Koller, 2001] ACTIVE-SVM's SIMPLE variant by replacing the SVM parts with a BPM. A noteworthy observation is that SIMPLE is known to fail for some supposedly difficult version space configurations, it appears that these difficult cases are instances where $CG(\mathcal{W})$ and $CC(\mathcal{W})$ are actually far from each other. In other words, SIMPLE is at its best when it approximates correctly our BPM based query procedure and thus fits the framework of our theoretical analysis, conversely its performances degrade when the two strategies start diverging, notably theory and experiment show that ACTIVE-BPM performs as usual in those cases.

**Algorithm 10** Top: The ACTIVE-BPM Cutting Planes active learning procedure; $\boldsymbol{w}^k$ is computed as the center of mass (i.e BPM) of $\mathcal{W}^k$ (line 15) but can replaced by any other classifier thus spanning an entire family of Cutting Pslanes learning procedures. Bottom: a possible implementation of QUERY() : sampling strategies are given in, e.g., [Herbrich et al., 2001, Lovász and Vempala, 2006, Kannan and Narayanan, 2012]

1: $\mathcal{W}^0 \leftarrow \mathcal{B}_1$
2: $k \leftarrow 0$
3: **repeat**
4: $\quad \boldsymbol{w}^k \leftarrow CG(\mathcal{W}^k)$
5: $\quad (\boldsymbol{x}_i, t_i) \leftarrow \text{QUERY}(\mathcal{W}^k, \mathcal{S})$
6: $\quad$ **if** $t_i \left\langle \boldsymbol{w}^k; \boldsymbol{x}_i \right\rangle < 0$ **then**
7: $\quad\quad \mathcal{W}^{k+1} \leftarrow \mathcal{C}^k \cap \{\boldsymbol{v} : t_i \left\langle \boldsymbol{v}; \boldsymbol{x}_i \right\rangle \geq 0\}$
8: $\quad\quad k \leftarrow k + 1$
9: $\quad$ **end if**
10: **until** $\mathcal{W}^k$ is small enough
11: **return** $\boldsymbol{w}^k$
12:
13: **function** QUERY($\mathcal{C}, \mathcal{S}$)
14: $\quad$ Sample $M$ points $\boldsymbol{x}_1, \dots \boldsymbol{x}_M$ from $\mathcal{W}$
15: $\quad \boldsymbol{v} \leftarrow \sum_{k=1}^{M} \boldsymbol{x}_k / M$
16: $\quad \boldsymbol{x} \leftarrow \arg\min_{\boldsymbol{x}_i \in \mathcal{S}} \left\langle \boldsymbol{v}; \boldsymbol{x}_i \right\rangle$
17: $\quad t \leftarrow$ get label from an expert
18: $\quad$ **return** $(\boldsymbol{x}, t)$
19: **end function**

## 7.4 Experimental results

We illustrate our method for active learning on text classification data. For easy comparison, we follow an experimental procedure similar as the one in [Tong and Koller, 2001]. Namely, we use the *Reuters-21578 —ModApte* variation— and Newsgroups datasets[1]. The Reuters dataset is composed of 8,293 documents represented in TF-IDF form for 18,933 words. The dataset spans 65 topics such as *Earn*, *Coffee* or *Cocoa* and is split in 5,946 training examples and 2,347 test examples. On the other hand, the Newsgroups dataset accounts for 18,846 documents of 26,214 features split in 20 topics. Half of this dataset is uniformly picked for training while the rest is kept for testing purposes. On both datasets we train a "one-versus-all" classifier for each class. We start by creating a pool of unlabeled training examples sampled from the training set. Then we run algorithm 10. We use two variations of the QUERY function: one based on the Chebyshev's center (note that this is equivalent to the ACTIVE-SVM of [Tong and Koller, 2001]), and the other based on an approximation of the center of gravity from Minka's Expectation Propagation method [Minka, 2001a]. This last approach corresponds to the ACTIVE-BPM algorithm and was notably proposed in [Brinker, 2004] independently. It is a direct application of Active Learning algorithms with Cutting planes

[1] Available at `http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html`

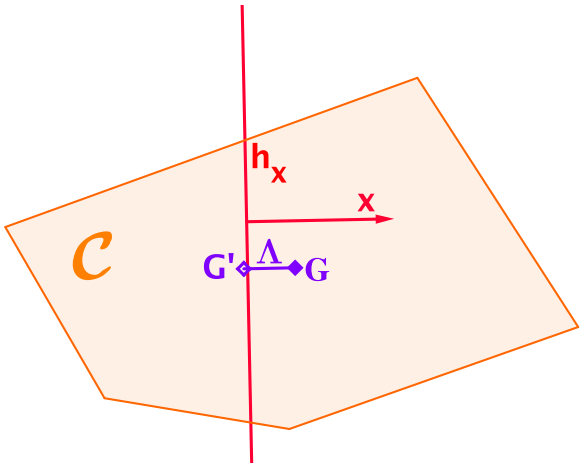



Figure 7.4 – *An illustration of the idea behind our convergence result. Given a version space $\mathcal{W}$ with center of gravity $G$, if the Active Learning Oracle returns a cut that does not exclude the center of gravity $G$ we will consider instead an approximate center of gravity —$G'$ here— such that $G'$ lies just at the limit of the cut. Measuring the distance $\Lambda$ between $G$ and $G'$ give a rough idea of how the the version space will be split according to theorem 5.2.*

method to the Bayes Point Machine. For both methods, we use two pools of different sizes (500 and 1,000 examples). For initialization reasons, each pool comes with two already labeled vectors[2].

All the computations are done with a linear kernel and the presented results are class-wise accuracy measurements on the test examples over the 10 most represented classes. The values reported here are an average of these measures over 25 runs. We complement these two datasets with Gunnar Raetsch's Banana dataset. The Banana dataset is a widely used bataset of 2-dimensionnal points split into two classes from which we extract 400 training and 4900 test examples. Due to its small size, the whole training set is used for the pool of unlabeled example. The computations are realized with an RBF kernel of parameter $\sigma = 0.5$ and presented results are averaged over 50 runs.

Figure 7.5 graphically depicts the behavior of the so-called ACTIVE-SVM [Tong and Koller, 2001] and the ACTIVE-BPM algorithms on each dataset. Namely, in both algorithms, the queries are selected according to their distance to the "centroid" of $\mathcal{W}^k$, which, in turn, serves as classifier. The difference between these two algorithms lies in that ACTIVE-SVM uses the Chebyshev center and ACTIVE-BPM the center of gravity for centroid. In figure 7.5, plots are represented by circles or squares whether they correspond to results achieved by ACTIVE-SVM or ACTIVE-BPM. Additionally, for the Reuters and Newgroups datasets, dashed plots correspond to the pool of 500 examples while dotted plots relate to the pool of 1000 examples. The error bounds on the third plot (Banana) shows the usual standard deviation. Each plot represents the accuracy of those algorithms with

[2]SVM and CC are computed with libSVM: http://www.csie.ntu.edu.tw/ cjlin/libsvm/. BPM and CG are computed from Minka's own implementation of EP for BPM in matlab: `http://research.microsoft.com/en-us/um/people/minka/papers/ep/bpm`. The motivation here is essentially the ease of use provided by Minka's code and result should not vary by much when using the regular Billiard algorithm proposed in [Herbrich et al., 2001, Rujan, 1997].

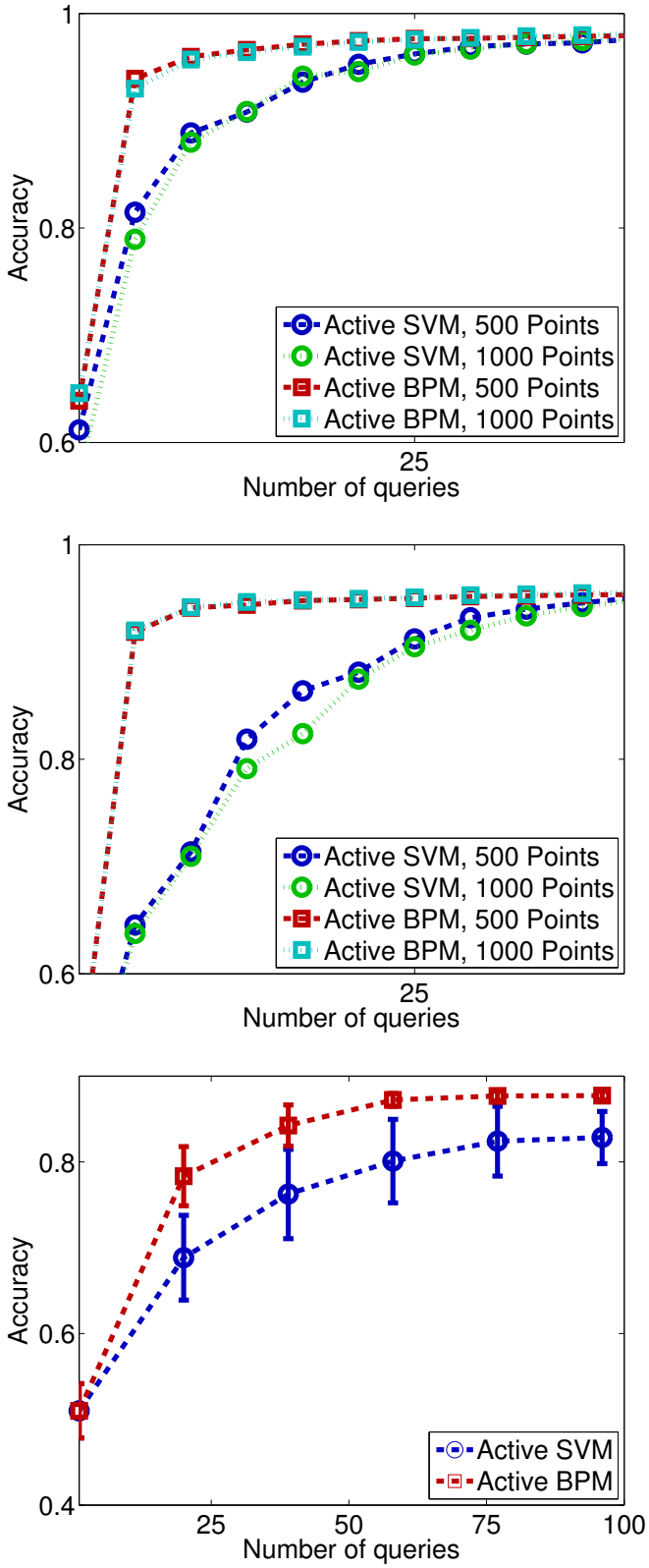



Figure 7.5 – *Accuracy on the Reuters (top) and Newsgroups (middle) datasets for* ACTIVE-SVM *and* ACTIVE-BPM *for pools of* 500 *and* 1000 *examples. Bottom: accuracy with error bars on the Banana dataset (Gunnar Raetsch) for* ACTIVE-SVM *and* ACTIVE-BPM.

respect to the number of queries made. We can see that ACTIVE-BPM systematically outperforms ACTIVE-SVM and increases its accuracy faster for all datasets, already attaining an accuracy of 0.9 after roughly 10 queries for both Reuters and Newsgroups datasets. Both algorithm seem to stabilize after 30 queries, with the ACTIVE-BPM being slightly more accurate than its SVM counterpart. For the Banana dataset, the accuracy increase in the first queries is a lot smoother, with an accuracy for ACTIVE-BPM of roughly 0.8 after 20 queries. Both algorithms seem to have converged after 60 queries. Comparatively, not only does ACTIVE-BPM clearly dominate its SVM counterpart but it is also more stable as evidenced by the error bars which become negligible past the $60^{\text{th}}$ query.

## 7.5 Conclusion

In this last chapter we have discussed how Active Learning and Cutting Planes settings are close. A focus point of our analysis was in establishing that both settings are build upon a common ground with only mild differences. Notably, our approach has revolved around building a Cutting Planes Oracle from an Active Learning Oracle and studying the convergence properties of the resulting Cutting Planes algorithm. In doing so, we made use of theorem 5.2 which was instrumental in establishing the efficacy of Cutting Planes methods in the Active Learning setting. The result is an Active Learning algorithm previously hinted by [Brinker, 2004] that build upon the Cutting Planes theory to devise a query strategy that combines the theoretical soundness of Expected Model change and the simplicity of Uncertainty sampling strategies. Additionally, our theoretical analysis provides a new insight on the results of [Tong and Koller, 2001] and the limitations of their SIMPLE algorithm. Finally, this chapter comes as yet another hint at the relevance of geometric methods for machine learning problems and in this sense it complements and reinforces the discussions we held in chapter 6.

# Conclusion

## Back and There Again

In this thesis, we have studied and discussed various machine learning methods that have in common to propose alternative ways to work around label unavailability in classification learning. Namely, contrary to the semi supervised setting, these methods focus on playing on label availability, either to make labelling easier, at the cost of label consistency, or by altering the supervised setting learning scheme in order to reduce the label requirement of learning. Specifically, we were interested in confused learning on one hand, that is learning under a class-wise characterized noise pattern, and active learning on the other hand, where labels are actively queried on the fly by the learning algorithm. Besides, we also provided a detailed theoretical discussion on the link between machine learning and localization methods that bore interesting results, both to computational geometry and regular supervised learning.

Our first contribution (chapter 4) was toward confused learning where we proposed to extend the works of [Blum et al., 1998, Bylander, 1994] for learning under confusion noise in a bi-class setting to the problem of multiclass classification. Namely in the setting of multiclass classification, the labels are assumed to be produced by a corrupted concept that differs from the target concept on some examples in a way that is class-wise constant; that is to say for each couple of class $(p, q)$ the probability for an example of true class $p$ to be labelled $q$ in the dataset, that is the confusion rate between $p$ and $q$, is constant. Moreover, this information is assumed to be known (or, at least, accurately estimated) and our algorithm relies on the use of the confusion matrix which regroups those aforementioned confusion rates for every couples of class $(p, q)$. The algorithm we proposed, called UMA algorithm (*Unconfused Multiclass Additive* algorithm), is able to achieve consistent learning with respect to the true, obfuscated, concept from the confused training set and the confusion matrix under the assumption of a invertible confusion matrix, which is, as we have discussed, a tolerable assumption in practice. Notably, UMA is based upon the general update scheme of the *Ultraconservative Additive* algorithms family introduced in [Crammer and Singer, 2003] which is both a generalization of the perceptron update scheme and an extension to multiclass problems. To the best of our knowledge, UMA is the first algorithm to learn under confusion noise in a multiclass setting. Additionally, we provided a detailed theoretical analysis of UMA that not only established its convergence properties from a formal point of view, but also touched upon some of the key elements of the algorithm. We also have discussed the specifics of UMA update scheme and how different tunings of the algorithm may lead to different behavior in practice and we tackled the question of making UMAto work with kernels. Finally we complemented our study with synthetic and real world experiments that backed our previous theoretical results on UMA.

Our second contribution (chapter 5) is actually more related to the

field of computational geometry than machine learning. Centers of gravity are known for their property to capture some interesting feature of their parent convex body, namely one of the most remarkable property of centers of gravity is that any hyperplane going through the center of gravity of a convex body splits it in two parts in such a way that the volume of the first is no less than a fraction $1/e$ of the second's (theorem 5.1, [Grunbaum, 1960]). Besides, a noteworthy peculiarity of this result is its independence from both the dimension of the convex body and the orientation of the hyperplane. It should come at no surprise that centers of gravity are notoriously hard to compute, and approaches based on centers of gravity's theory tend to rely in practice to approximated centroids, those being either based on geometric arguments such as the Chebyshev's center or numerical approximation through the use of sampling methods. Yet, the question of the partitioning properties of those approximations is mostly unknown from a theoretical point of view and despite strong empirical evidence of the soundness of such approaches, we are not aware of any result on how theorem 5.1 degrades when approximated centers of gravity are used. We proposed theorem 5.2 which partly answer this question and directly link the partition's volume ratio with the distance between the approximated centroid and the true center of gravity. On the other hand, we lose the attractive independence in the dimension and hyperplane's orientation we had in theorem 5.1 although this had to be expected given the nature of the result. Our theorem is a direct extension of theorem 5.1 and its proof, which is given in full in appendix A.3, is a non trivial extension of the original proof of theorem 5.1 found in [Grunbaum, 1960]. Also note that for the sake of completeness a rewriting of the proof presented originally in [Grunbaum, 1960] is given in appendix A.3.2. The applications of theorem 5.2 might be multiple and although its use in this thesis served machine learning purposes we shall emphasize that it is ultimately a result that have relevance beyond the field of machine learning and particularly in computational geometry.

The penultimate contribution of this work (chapter 6) is essentially in the form of a new supervised algorithm based on the Perceptron. The Perceptron algorithm is a seminal algorithm of machine learning that trade the compression properties of modern learning methods such as SVMs for an elegant and online update scheme. More importantly though, one of the greatest strength of the Perceptron comes from its convergence property (theorem 1.3). More precisely, a Perceptron algorithm will eventually converge in a time independent of the number of examples, as long as those examples are linearly separable. Our contribution is based on two observations: first, a Perceptron will converge even though it is fed with artificial example designed specifically to achieve good compression as long as they are separable and second, that Cutting Planes methods naturally embed compression scheme features because of their oracle query scheme that ensure an efficient and simplified reconstruction of the version space. The algorithm we have proposed leverages the strengths of both methods and as such it enjoys a convergence rate on par with a Perceptron but find a solution that depends on a lot less examples which induces some improved generalization guarantees from a Compression Scheme standpoint. In addition synthetic experimental result allowed us

to validate our theory and improve on the theoretical bound by exhibiting a speed up in the number of updates relatively to a regular Perceptron. Beyond the algorithmic interest of this contribution, it is a testimony on how machine learning can benefit from a more geometric interpretation of its fundamental problems and it is our belief that this contribution is merely one among the many links one can draw between machine learning and localization methods.

Finally our last contribution (chapter 7) caught up with the recurring theme of this thesis, playing with label availability and concerned the setting of Active Learning where all the data are unlabeled at first and the labels are actively queried by the learning algorithm depending of their expected relevance. Recent results in Active Learning suggest that the active query scheme allows for an important reduction in the number of labels required with respect to supervised learning at similar accuracy rates [Settles, 2010]. One of the fundamental work of the past years in this domain is [Tong and Koller, 2001] which proposed the SIMPLE active algorithm and demonstrated its applicability in text classification problems. Notably, SIMPLE takes its ground on a geometric approach of active learning that is strongly reminiscent of Cutting Planes methods even though it is never presented as such in the original paper, moreover the authors argue that part of the success of their algorithm is because the SVM classifier is the Chebyshev center of the learning problem's version space, that is the set of all consistent classifier with the labelled data. Our contribution was to build on their idea and formally study the problem of Active Learning through the geometric point of view we have discussed in chapters 5 and 6. The crux of our contribution was to argue that Cutting Planes theory provides a suitable framework to study Active Learning algorithms in the sense that the two settings are built upon the same ideas and algorithmic schemes. Notably, we propose to base the active queries on the version's space center of gravity rather than Chebyshev's center as in ACTIVE-SVM algorithm. According to theorem 5.2 we argued that this is a theoretically sound strategy. Moreover we showed that theorem 5.2 helps explaining the drop in efficacy observed in ACTIVE-SVM when the Chebyshev's and gravity centers are far away. Although our contribution focused on learning Bayes Point Machine classifiers because of their natural connection to centers of gravity, the active learning query scheme we proposed can be used to turn many passive learning algorithms into active ones. Finally, it is worth noting that our contribution acts as a middle ground between the two opposed Active Learning paradigms that are Expected Model change and Uncertainty Sampling, in particular, center of gravity based queries allow for the computationally efficient query scheme of uncertainty sampling methods as well as the version space's halving property of Expected Model chance approaches.

**Publications**

The contributions of this thesis have been previously published in peer-reviewed conferences and journals. The results of chapter 4 were first presented at the *Asian Conference on Machine Learning* in 2013 [Louche and Ralaivola, 2013] and an extended revision was also published

in *Machine Learning Journal* [Louche and Ralaivola, 2015b]. The contributions we discussed in chapter 5, 6 and 7 were presented together in the single work at the *International Conference on Neural Network* in 2015 [Louche and Ralaivola, 2015a] which won the *best student paper award*.

## PERSPECTIVES

Beyond the immediate interest of the contributions presented therein it should also be discussed the future research avenues our results have opened and on many aspects this thesis is nothing but a little step toward more efficient learning methods for confused and active learning, as well as a better understanding of classification from a geometric perspective.

On the matter of learning under confusion noise, the immediate question that our work prompts is the one of extending our model to other noise models than confusion noise and we think that an argument along the line of [Blum et al., 1998, Bylander, 1994] can be used to back the relevance of confusion noise to other, more local, settings. Also, one of the current drawback of UMA is the invertibility requirement over the confusion matrix which seems to be a limiting assumption and a possible improvement for UMA would be to replace the invertibility assumption with a less restricting criterion. An other interesting observation about UMA is that it was found to handle quasi-linear problem such as Reuters quite well and a possible continuation for this work would be to study how learning can be achieved under confusion noise for nearly linear target concepts. More precisely, it seems that the capacity of UMA to cope with confused dataset might allow to handle a little overstrain on the hypothesis class.

In addition, the last part of this thesis opens a lot of questions because of its more exploratory nature and leaves a lot of points to be investigated. In the first place, it is the general connection between machine learning, and more particularly classification, and localization methods that is left to be explored in greater details. One interesting observation we had during our work was that Cutting Plane theory and machine learning developed the same solutions to similar problem through the use of centers of gravity and Chebyshev centers, and the link between the two domains would have been explicit sooner it would have allowed for a better understanding of machine learning. A point of interest for future research is to study how more complex machine learning settings (e.g. multi-task learning) may benefit from a similar geometrical interpretation.

On a less general level, it seems to us that there is more to the problem of approximate centers of gravity than what we have presented in theorem 5.2 and our result is still very dependent on the shape of the parent convex body, the direction of the hyperplane and the dimension. In the future, an interesting path to follow is to investigate similar results that do not depend on the distance between the center of gravity and its approximation but on more geometric considerations in the hope that it would help produce a more general result. Another interesting question we would like to investigate is the relevance of centerpoints to these questions and how they could relate to machine learning theory.

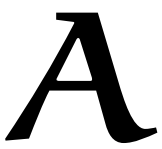
The modified Perceptron algorithm we propose would also benefit from a more thorough study and could be extended in numerous ways. First, it would be interesting to properly asserts its compressions properties in practical settings, moreover, we lack a theoretical result on the number of Cutting Planes iteration performed (beyond the general bound given by the convergence theorem) while our experiment shows that very few cuts are getting queried in practice. Another interesting improvement would be for the implementation of a margin optimization scheme into the algorithm based on the margin's gradient, much like how Cutting Planes methods are used in quadratic optimization; this would notably allow for the algorithm to fit the large margin paradigm of SVM methods.

More generally, a practically appealing research path would be to modify Cutting-Planes based learning methods such as our modified Perceptron algorithm to make them robust to noise in a similar than [Joachims et al., 2009] for SVMs. This is especially relevant to the BPM algorithm that has yet to be properly studied within a Cutting Planes framework, although it seems that the algorithm would greatly benefit from this approach because of the cost incurred for the computation of the center of gravity and the lack of compression scheme result for regular BPM. Moreover a Cutting Planes query scheme applied to the BPM algorithm would fully take advantage of the BPM classifier being the center of gravity of the version space and allow for strong theoretical compression guarantees.

Finally, the Active Learning query strategy we propose may be potentially improved in a number of ways. Notably, our query strategy only take into account the distance of a data hyperplane to the version space's center of gravity and as such we completely ignore the hyperplane orientation. Given the result of theorem 5.2 it seems reasonable to think that the orientation of the data's hyperplane should be relevant in the query strategy and it is one of the most immediate extension of our work. Another interesting potential improvement comes from the fact that we do not re-use the billiard trajectory computed during the center of gravity estimation procedure of our algorithm, namely this sampling could be leveraged to give a rough estimation of the version's space volume after each possible cut, this would notably allow to detect some limit case where queries close to the center of gravity may not yield the expected volume reduction. Also, note that a better exploitation of theorem 5.2 could also lead to a smarter query strategy where some of the suboptimal queries could be easily discarded regardless of their distance to the center of gravity. Lastly, we may mention that besides centers of gravity there exists other approximate centroids, the Analytic center being one particular example, that may allow for theoretical guarantees in an active learning context without the computational burden of center of gravity estimation.

To conclude we would emphasize that our active learning strategy is an application of the results presented in chapters 5 and to a lesser extent chapter 6, yet there is more to these chapters than our contribution to Active Learning. A general perspective and ambitious extension of this thesis is to provide a comprehensive study of geometrical methods applied to machine learning.

# Appendix



Contents



## A.1 Appendix of Part I

In this section we will present the proof to the Ultraconservative convergence theorem (theorem 2.3). The proof is essentially a rewriting of the one found in [Crammer and Singer, 2003] with the set of notations we used in this thesis. Yet, we feel like this proof is fundamental with respect to the present work and should nonetheless be included here. Part of this is because Ultraconservative algorithms are central to our work, not only because of their relevance to Part II but also because Rosenblatt's Perceptron algorithm (and most of its variants) ultimately falls within the framework of ultraconservative additive algorithms. Namely, the Alg. 1 and its variations introduced in this work can be rewritten as an additive algorithm where $Q = 1$ —note that in this case $Q = 1$ and $Q = 2$ are equivalent since in the latter case $\boldsymbol{w}_1 = -\boldsymbol{w}_2$ is enforced by the additive update scheme. More importantly, theorem 2.3 is a generalization of the result of Block and Novikoff (theorem 1.3) and the two results are equivalent for $Q = 1$. Finally, the proof we will present is also a generalization of the proof of theorem 1.3 and applying the exact same reasoning to the setting of a bi-class Perceptron yields the Block-Novikoff theorem; this fact also extends to most variants of the Perceptron algorithm hence this proof should be considered as a general purpose proof technique for any algorithm derived from a Perceptron's update scheme and more particularly to anything we have exposed so far that is related to the Perceptron's algorithm (Alg. 1, 4, 6).

### A.1.1 Proof of Theorem 2.3

We first remind the fundamental set of constraint over the update vectors that define the family of ultraconservative algorithm. This will serve as an easy reference since the proof is constructed upon this set of constraints, for more detail we refer the reader to the full algorithm (Alg. 2) in section 2.3.2.

**Remark A.1** (Update Constraints for the Ultraconservative Additive Algorithm family) *Any algorithm belonging to the* Ultraconservative Additive *algorithm family has the following update rule:*

$$\forall r \in [Q] : \mathbf{W}^k_{\cdot r} \leftarrow \mathbf{W}^k_{\cdot r} + \tau^k_r \boldsymbol{x}_i$$

*where $\mathbf{W}^k$ is the current (multiclass) classifier at step $k$, $\boldsymbol{x}_i$ a data on which $\mathbf{W}^k$ errs, $Q$ the number of class and $\tau^k \in \mathbb{R}^Q$ a real vector compliant with the following rules:*

$$\sum_{r=1}^{Q} \tau^k_r = 0 \tag{A.1}$$

$$\tau^k_{t_i} = 1 \tag{A.2}$$

$$r \notin \mathcal{E} \cup \{t_i\} \Rightarrow \tau^k_r = 0 \tag{A.3}$$

**Theorem A.1** (Mistakes Bound For Ultraconservative Additive Algorithm) *Let $(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)$ be any sequence of points in $\mathbb{X} \times \mathbb{Y}$ such that $\|\boldsymbol{x}_i\| \leq R$ for all $i$. If there exists a multiclass classifier $\mathbf{W}^*$ of norm $\|\mathbf{W}^*\|_F$ such that*

$$\gamma^{\mathbf{W}^*} \doteq \min_{i \in [N]} \{ \langle \mathbf{W}^*_{\cdot t_i}; \boldsymbol{x}_i \rangle - \max_{r \neq t_i} \langle \mathbf{W}^*_{\cdot r}; \boldsymbol{x}_i \rangle \}$$

*then, any Ultraconservative Additive Algorithm make at most*

$$\frac{2R^2}{(\gamma^{\mathbf{W}^*})^2}$$

*updates and output a classifier $\mathbf{W}$ that make no mistake on the sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$.*

As a preliminary, we will need the following lemma in order to ease the unfolding of the proof.

**Lemma A.1** *For any vector $\tau$ following the rules of remark A.1 the following holds true:*

$$\sum_{r=1}^{Q} \tau_r^2 \leq 2\tau_{t_i} \leq 2$$

*proof of lemma A.1:* From the aforementioned rules, we have that

$$\begin{aligned} \sum_{r=1}^{Q} \tau_r &= \sum_{r \neq t_i} \tau_r + \tau_{t_i} \\ &= \tau_{t_i} - \tau_{t_i} \qquad \text{(first rule)} \end{aligned}$$

Hence,

$$\|\tau\|_1 = 2\tau_{t1} \quad \text{and} \quad \|\tau\|_\infty = \tau_{t_i}$$

Applying Hölder's inequality we get:

$$\begin{aligned}\sum_{k=1}^{Q}\tau_r^2 &= \sum_{k=1}^{Q}(\tau_r\tau_r)\\ &= \|\tau\tau\|_1\\ &\leq \|\tau\|_1\|\tau\|_\infty && \text{(Hölder's inequality)}\\ &= 2\tau_{\tau_i}\tau_{t_i}\\ &\leq 2\tau_{t_i} \leq 2\end{aligned}$$

□

*Proof of theorem 2.3.* Let assume that the algorithm has already made $k$ updates. Because an update is only computed when an error occurs, that is to say that the algorithm has already made $k$ error during the training phase. We will write $\mathbf{W}^{k+1}$ the current classifier after the $k$th update, that is $\forall r \in [Q] : \mathbf{W}^{k+1}_{\cdot r} = \mathbf{W}^{k}_{\cdot r} + \tau^k_r + \boldsymbol{x}^k_{\text{up}}$ where $\boldsymbol{x}^k_{\text{up}}$ denote the update vector $\boldsymbol{x}_i$ of class $t_i$ on which $\mathbf{W}^k$ has made an error (see also Alg. 2. The proof will proceed by bounding $\|\mathbf{W}^{k+1}\|^2_{\text{F}}$ from above and below, a by-product of doing so is that this will allows for $k$ to also be bounded from above —that is the number of update the algorithm can possibly make.

First, we derive the lower bound on $\|\mathbf{W}^{k+1}\|^2_{\text{F}}$, this can be done by bounding the term $\sum_{r=1}^{Q}\left\langle \mathbf{W}^*_{\cdot r}; \mathbf{W}^{k+1}_{\cdot r}\right\rangle$ with respect to $\sum_{r=1}^{Q}\left\langle \mathbf{W}^*_{\cdot r}; \mathbf{W}^{k}_{\cdot r}\right\rangle$. In other words, we are interested in how the current classifier get closer to $\mathbf{W}^*$ with every updates. Namely

$$\begin{aligned}\sum_{r=1}^{Q}\left\langle \mathbf{W}^*_{\cdot r}; \mathbf{W}^{k+1}_{\cdot r}\right\rangle &= \sum_{r=1}^{Q}\left\langle \mathbf{W}^*_{\cdot r}; \mathbf{W}^{k}_{\cdot r} + \tau^k_r\boldsymbol{x}^k_{\text{up}}\right\rangle\\ &= \sum_{r=1}^{Q}\left\langle \mathbf{W}^*_{\cdot r}; \mathbf{W}^{k}_{\cdot r}\right\rangle + \sum_{r=1}^{Q}\tau^k_r\left\langle \mathbf{W}^*_{\cdot r}; \boldsymbol{x}^k_{\text{up}}\right\rangle \qquad \text{(A.4)}\end{aligned}$$

Focusing on the second term on the right we have that

$$\begin{aligned}\sum_{r=1}^{Q}\tau^k_r\left\langle \mathbf{W}^*_{\cdot r}; \boldsymbol{x}^k_{\text{up}}\right\rangle &= \sum_{r\neq t_i}\tau^k_r\left\langle \mathbf{W}^*_{\cdot r}; \boldsymbol{x}^k_{\text{up}}\right\rangle + \tau^k_{t_i}\left\langle \mathbf{W}^*_{\cdot t_i}; \boldsymbol{x}^k_{\text{up}}\right\rangle\\ &= \sum_{r\neq t_i}\tau^k_r\left\langle \mathbf{W}^*_{\cdot r}; \boldsymbol{x}^k_{\text{up}}\right\rangle - \left(\sum_{r\neq t_i}\tau^k_r\right)\left\langle \mathbf{W}^*_{\cdot t_i}; \boldsymbol{x}^k_{\text{up}}\right\rangle\\ &\qquad\qquad \text{(From } \tau_{t_i} = -\textstyle\sum_{r\neq t_i}\tau_r \text{ (rule (A.1)))}\\ &= \sum_{r\neq t_i}\tau^k_r\left\langle \mathbf{W}^*_{\cdot r}; \boldsymbol{x}^k_{\text{up}}\right\rangle - \left\langle \mathbf{W}^*_{\cdot t_i}; \boldsymbol{x}^k_{\text{up}}\right\rangle\\ &= \sum_{r\neq t_i} -\tau^k_r\left(\left\langle \mathbf{W}^*_{\cdot t_i}; \boldsymbol{x}^k_{\text{up}}\right\rangle - \left\langle \mathbf{W}^*_{r}; \boldsymbol{x}^k_{\text{up}}\right\rangle\right)\end{aligned}$$

Using the assumption that $\mathbf{W}^*$ realizes a margin $\gamma^{\mathbf{W}^*}$, that is, for any $r \in [Q]$ then $\left\langle \mathbf{W}^*_{\cdot t_i}; \boldsymbol{x}_i\right\rangle - \left\langle \mathbf{W}^*_{\cdot r}; \boldsymbol{x}_i\right\rangle \geq \gamma^{\mathbf{W}^*}$ we have that

$$\begin{aligned}\sum_{r\neq t_i} -\tau_r^k \left(\left\langle \mathbf{W}_{\cdot t_i}^*; \boldsymbol{x}_{\text{up}}^k \right\rangle - \left\langle \mathbf{W}_r^*; \boldsymbol{x}_{\text{up}}^k \right\rangle\right) &\geq \sum_{r\neq t_i} -\tau_r^k \gamma^{\mathbf{W}^*} \\ &= \tau_{t_1}^k \gamma^{\mathbf{W}^*} \qquad \text{(First rule (A.2))} \\ &= \gamma^{\mathbf{W}^*} \qquad \text{(second rule (A.2))}\end{aligned}$$

Hence we have that

$$\sum_{r=1}^{Q} \tau_r^k \left\langle \mathbf{W}_{\cdot r}^*; \boldsymbol{x}_{\text{up}}^k \right\rangle \geq \gamma^{\mathbf{W}^*} \tag{A.5}$$

Putting this back into equation (A.4) we get

$$\begin{aligned}\sum_{r=1}^{Q} \left\langle \mathbf{W}_{\cdot r}^*; \mathbf{W}_{\cdot r}^{k+1} \right\rangle &= \sum_{r=1}^{Q} \left\langle \mathbf{W}_{\cdot r}^*; \mathbf{W}_{\cdot r}^{k} \right\rangle + \sum_{r=1}^{Q} \tau_r^k \left\langle \mathbf{W}_{\cdot r}^*; \boldsymbol{x}_{\text{up}}^k \right\rangle \\ &\geq \sum_{r=1}^{Q} \left\langle \mathbf{W}_{\cdot r}^*; \mathbf{W}_{\cdot r}^{k} \right\rangle + \gamma^{\mathbf{W}^*}\end{aligned}$$

Recursively applying the same calculations on $\sum_{r=1}^{Q} \left\langle \mathbf{W}_{\cdot r}^*; \mathbf{W}_{\cdot r}^{k} \right\rangle$ and for each of the $k$ previous classifiers we obtain that

$$\sum_{r=1}^{Q} \left\langle \mathbf{W}_{\cdot r}^*; \mathbf{W}_{\cdot r}^{k+1} \right\rangle \geq k\gamma^{\mathbf{W}^*} \tag{A.6}$$

Finally, from the definition of the Frobenius norm:

$$\begin{aligned}\|\mathbf{W}^*\|_{\text{F}}^2 \|\mathbf{W}^{k+1}\|_{\text{F}}^2 &= \left(\sum_{r=1}^{Q} \|\mathbf{W}_{\cdot r}^*\|_2^2\right)\left(\sum_{r=1}^{Q} \|\mathbf{W}_{\cdot r}^{k+1}\|_2^2\right) \\ &\geq \left(\sum_{r=1}^{Q} \|\mathbf{W}_{\cdot r}^*\|_2 \|\mathbf{W}_{\cdot r}^{k+1}\|_2\right)^2 \qquad \text{(Cauchy-Schwartz)} \\ &\geq \left(\sum_{r=1}^{Q} \left\langle \mathbf{W}_{\cdot r}^*; \mathbf{W}_{\cdot r}^{k+1} \right\rangle\right)^2 \\ &\qquad \text{(Cauchy-Schwartz on each sub-temrs)} \\ &\geq \left(k\gamma^{\mathbf{W}^*}\right)^2\end{aligned}$$

Using the assumption that $\|\mathbf{W}^*\|_{\text{F}} = 1$ yield the lower bound

$$\|\mathbf{W}^{k+1}\|_{\text{F}}^2 = \|\mathbf{W}^*\|_{\text{F}}^2 \|\mathbf{W}^{k+1}\|_{\text{F}}^2 \geq \left(k\gamma^{\mathbf{W}^*}\right)^2 \tag{A.7}$$

Next, we bound $\|\mathbf{W}^{k+1}\|_{\text{F}}$ from above. Re-using the previous notations, we have:

$$\begin{aligned}
\|\mathbf{W}^{k+1}\|_{\mathrm{F}} &= \sum_{r=1}^{Q} \|\mathbf{W}_{\cdot r}^{k+1}\|_2^2 \\
&= \sum_{r=1}^{Q} \|\mathbf{W}_{\cdot r}^{k} + \tau_r^k \boldsymbol{x}_{\mathrm{up}}^k\|_2^2 \\
&= \sum_{r=1}^{Q} \|\mathbf{W}_{\cdot r}^{k}\|_2^2 + 2\sum_{r=1}^{Q} \tau_r^k \left\langle \mathbf{W}_{\cdot r}^{k}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle + \sum_{r=1}^{nbClass} \left(\tau_r^k\right)^2 \|\boldsymbol{x}_{\mathrm{up}}^k\|_2^2 \\
&= \|\mathbf{W}_{\cdot r}^{k}\|_{\mathrm{F}}^2 + \|\boldsymbol{x}_{\mathrm{up}}^k\|_2^2 \sum_{r=1}^{Q} \left(\tau_r^k\right)^2 + 2\sum_{r=1}^{Q} \tau_r^k \left\langle \mathbf{W}_{\cdot r}^{k}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle \qquad \text{(A.8)}
\end{aligned}$$

As previously in Eq. (A.4) the last term can be expended as follow

$$2\sum_{r=1}^{Q} \tau_r^k \left\langle \mathbf{W}_{\cdot r}^{k}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle = 2\sum_{r \neq t_i} -\tau_r^k \left( \left\langle \mathbf{W}_{\cdot t_i}^{*}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle - \left\langle \mathbf{W}_{r}^{*}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle \right) \qquad \text{(A.9)}$$

Two cases are possible the sum above, depending on $r$. Either $\left\langle \mathbf{W}_{\cdot t_i}^{*}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle - \left\langle \mathbf{W}_{r}^{*}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle \geq 0$ and $r \notin \mathcal{E}$, thus from the third rule (A.2) $\tau_r^k = 0$; or $\left\langle \mathbf{W}_{\cdot t_i}^{*}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle - \left\langle \mathbf{W}_{r}^{*}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle < 0$ and from the first rule (A.1) we know that $\tau_r^k \leq 0$. As a result, the term is the sum of sub-terms each negative or null therefore

$$2\sum_{r \neq t_i} -\tau_r^k \left( \left\langle \mathbf{W}_{\cdot t_i}^{*}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle - \left\langle \mathbf{W}_{r}^{*}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle \right) \leq 0 \qquad \text{(A.10)}$$

Putting this back into equation (A.8) gives

$$\begin{aligned}
\|\mathbf{W}^{k+1}\|_{\mathrm{F}} &= \|\mathbf{W}_{\cdot r}^{k}\|_{\mathrm{F}}^2 + \|\boldsymbol{x}_{\mathrm{up}}^k\|_2^2 \sum_{r=1}^{Q} \left(\tau_r^k\right)^2 + 2\sum_{r=1}^{Q} \tau_r^k \left\langle \mathbf{W}_{\cdot r}^{k}; \boldsymbol{x}_{\mathrm{up}}^k \right\rangle \\
&\leq \|\mathbf{W}_{\cdot r}^{k}\|_{\mathrm{F}}^2 + \|\boldsymbol{x}_{\mathrm{up}}^k\|_2^2 \sum_{r=1}^{Q} \left(\tau_r^k\right)^2 \\
&\leq \|\mathbf{W}_{\cdot r}^{k}\|_{\mathrm{F}}^2 + 2\|\boldsymbol{x}_{\mathrm{up}}^k\|_2^2 \qquad \text{(lemma A.1)} \\
&\leq \|\mathbf{W}_{\cdot r}^{k}\|_{\mathrm{F}}^2 + 2R \qquad \text{(Def of } R\text{)}
\end{aligned}$$

Once again, recursively applying the same calculation on $\mathbf{W}^k$ yields the desired bound:

$$\|\mathbf{W}_{\cdot r}^{k+1}\|_{\mathrm{F}}^2 \leq 2kR^2 \qquad \text{(A.11)}$$

Combining the two bound together we have that

$$\begin{aligned}
& k^2 \left(\gamma^{\mathbf{W}^*}\right)^2 \leq \|\mathbf{W}^{k+1}\|_{\mathrm{F}}^2 \leq 2kR^2 \\
\Leftrightarrow & k^2 \left(\gamma^{\mathbf{W}^*}\right)^2 \leq 2kR^2 \\
\Leftrightarrow & k \left(\gamma^{\mathbf{W}^*}\right)^2 \leq 2R^2
\end{aligned}$$

Therefore

$$k \leq \frac{2R^2}{\left(\gamma^{\mathbf{W}^*}\right)^2} \tag{A.12}$$

□

## A.2 Appendix of Part II

### A.2.1 Proof of Proposition 4.3

As a reminder we will restate proposition 4.3 like it was given in chapter 4.

**Proposition A.1** *Let $\varepsilon > 0$ and $\delta \in (0; 1]$. There exists a number*

$$N_0(\varepsilon, \delta, d, Q) = \mathcal{O}\left(\frac{1}{\varepsilon^2}\left[\ln\frac{1}{\delta} + \ln Q + d\ln\frac{1}{\varepsilon}\right]\right)$$

*such that if the number of training samples is greater than $N_0$ then, with high probability*

$$\left\langle \mathbf{W}^*_{\cdot q}; \boldsymbol{x}^{pq}_{up} \right\rangle - \left\langle \mathbf{W}^*_{\cdot k}; \boldsymbol{x}^{pq}_{up} \right\rangle \geq \gamma - \varepsilon \tag{A.13}$$

$$\left\langle W_{\cdot p}; \boldsymbol{x}^{pq}_{up} \right\rangle - \left\langle \mathbf{W}_{\cdot k}; \boldsymbol{x}^{pq}_{up} \right\rangle \geq 0 : \forall k \neq p. \tag{A.14}$$

The proof of this proposition can be found directly in the related section of Chapter 4 however the proof of the existence of $N_0$ was deferred to this appendix. The proof basically rely on the double sampling technique as it is discussed in [Devroye et al., 1996] but adapted to the specifics of our setting.

*Existence of $N_0$ (Prop. 4.3).* For a fixed pair $(p, q) \in \mathbb{Y}^2$, we consider the family of functions

$$\mathcal{F}_{pq} \doteq \{f : f(\boldsymbol{x}) \doteq \langle \boldsymbol{w}_q - \boldsymbol{w}_p; \boldsymbol{x}\rangle : \boldsymbol{w}_p, \boldsymbol{w}_q \in \mathcal{B}\}$$

where $\mathcal{B}$ is a $D$-dimensional unit ball where $D$ is the dimensionality of $\mathbb{X}$, as usual. For each $f \in \mathcal{F}_{pq}$ define the corresponding "loss" function

$$l^f(\boldsymbol{x}) \doteq l(f(\boldsymbol{x})) \doteq 2 - f(\boldsymbol{x})$$

Strictly speaking, $l^f(\boldsymbol{x})$ is not a loss as it does not take the predicted label $t(\boldsymbol{x})$ into account, nonetheless it does play the same role in the following proof than a classic loss in the regular double-sampling proof. One way to think of it is as the loss of a problem for which we do not care about the observed labels but instead we want to classify points into a predetermined class—in this case $q$.

Clearly, $\mathcal{F}_{pq}$ is a subspace of affine functions, thus $\text{Pdim}(\mathcal{F}_{pq}) \leq (D + 1)$, where $\text{Pdim}(\mathcal{F}_{pq})$ is the pseudo-dimension of $\mathcal{F}_{pq}$. Additionally, $l$ is Lipschitz in its first argument with a Lipschitz factor of $L \doteq 1$. Indeed $\forall t_0, t_1, t_2, \in \mathbb{Y} : |l(t_1, t_0) - l(t_2, t_0)| = |t_1 - t_2|$.

Let $\mathscr{D}_{pq}$ be any distribution over $\mathbb{X} \times \mathbb{Y}$ and $\mathcal{S} \in (\mathbb{X} \times \mathbb{Y})^M$ such that $\mathcal{S} \overset{\text{i.i.d}}{\sim} \mathscr{D}_{pq}^M$, then define the *empirical loss* $\text{err}_{\mathcal{S}}^l(f) \doteq \frac{1}{M} \sum_{\boldsymbol{x}_i \in \mathcal{S}} l(\boldsymbol{x}_i, t_i)$ and the *expected loss* $\text{err}_{\mathscr{D}_{pq}}^l(f) \doteq \mathbb{E}_{\mathscr{D}_{pq}}[l(\boldsymbol{x}, t)]$

The goal here is to prove that

$$\mathbb{P}_{\mathcal{S} \overset{\text{i.i.d}}{\sim} \mathscr{D}_{pq}^M} \left( \sup_{f \in \mathcal{F}_{pq}} |\text{err}_{\mathscr{D}_{pq}}^l[f] - \text{err}_{\mathcal{S}}^l[f]| \geq \epsilon \right) \in \mathcal{O}\left( \left( \frac{8}{\epsilon} \right)^{(D+1)} e^{M\epsilon^2/128} \right) \tag{A.15}$$

*Proof of* (A.15). We start by noting that $l(t_1, t_2) \in [0, 2]$ and then proceed with a classic 4-step double sampling proof. Namely:

**Symmetrization.** We introduce a *ghost* sample $\mathcal{S}' \in (\mathbb{X} \times \mathbb{Y})^M$, $\mathcal{S}' \overset{\text{i.i.d}}{\sim} \mathscr{D}_{pq}^M$ and show that for $f_{\mathcal{S}}^{\text{bad}}$ such that $|\text{err}_{\mathscr{D}_{pq}}^l[f_{\mathcal{S}}^{\text{bad}}] - \text{err}_{\mathcal{S}}^l[f_{\mathcal{S}}^{\text{bad}}]| \geq \epsilon$ then

$$\mathbb{P}_{\mathcal{S}'|\mathcal{S}} \left( \left| \text{err}_{\mathcal{S}'}^l[f_{\mathcal{S}}^{\text{bad}}] - \text{err}_{\mathscr{D}_{pq}}^l[f_{\mathcal{S}}^{\text{bad}}] \right| \leq \frac{\epsilon}{2} \right) \geq \frac{1}{2}$$

as long as $M\epsilon^2 \geq 32$.

It follows that

$$\begin{aligned}
&\mathbb{P}_{(\mathcal{S}, \mathcal{S}') \sim \mathscr{D}_{pq}^M \times \mathscr{D}_{pq}^M} \left( \sup_{f \in \mathcal{F}_{pq}} |\text{err}_{\mathcal{S}}^l[f] - \text{err}_{\mathcal{S}'}^l[f]| \geq \frac{\epsilon}{2} \right) \\
&\geq \mathbb{P}_{\mathcal{S} \sim \mathscr{D}_{pq}^M} \left( |\text{err}_{\mathcal{S}}^l[f_{\mathcal{S}}^{\text{bad}}] - \text{err}_{\mathscr{D}_{pq}}^l[f_{\mathcal{S}}^{\text{bad}}]| \geq \epsilon \right) \times \mathbb{P}_{\mathcal{S}'|\mathcal{S}} \left( \left| \text{err}_{\mathcal{S}'}^l[f_{\mathcal{S}}^{\text{bad}}] - \text{err}_{\mathscr{D}_{pq}}^l[f_{\mathcal{S}}^{\text{bad}}] \right| \leq \frac{\epsilon}{2} \right) \\
&\geq \frac{1}{2} \mathbb{P}_{\mathcal{S} \sim \mathscr{D}_{pq}^M} \left( |\text{err}_{\mathcal{S}}^l[f_{\mathcal{S}}^{\text{bad}}] - \text{err}_{\mathscr{D}_{pq}}^l[f_{\mathcal{S}}^{\text{bad}}]| \geq \epsilon \right) \\
&= \frac{1}{2} \mathbb{P}_{\mathcal{S} \sim \mathscr{D}_{pq}^M} \left( \sup_{f \in \mathcal{F}_{pq}} |\text{err}_{\mathcal{S}}^l[f] - \text{err}_{\mathscr{D}_{pq}}^l[f]| \geq \epsilon \right) \qquad \text{(By definition of } f_{\mathcal{S}}^{\text{bad}}\text{)}
\end{aligned}$$

Thus upper bounding the desired probability by

$$2 \times \mathbb{P}_{(\mathcal{S}, \mathcal{S}') \sim \mathscr{D}_{pq}^M \times \mathscr{D}_{pq}^M} \left( \sup_{f \in \mathcal{F}_{pq}} |\text{err}_{\mathcal{S}}^l[f] - \text{err}_{\mathcal{S}'}^l[f]| \geq \frac{\epsilon}{2} \right) \tag{A.16}$$

**Swapping Permutations.** Let define $\Gamma_M$ the set of all permutations that swap one or more elements of $\mathcal{S}$ with the corresponding element of $\mathcal{S}'$ (i.e. the $i$th element of $\mathcal{S}$ is swapped with the $i$th element of $\mathcal{S}'$). It is quite immediate that $|\Gamma_M| = 2^M$. For each permutation $\sigma \in \Gamma_M$ we note $\sigma(\mathcal{S})$ (resp. $\sigma(\mathcal{S}')$) the set originating from $\mathcal{S}$ (resp. $\mathcal{S}'$) from which the elements have been swapped with $\mathcal{S}'$ (resp. $\mathcal{S}$) according to $\sigma$.

Thanks to $\Gamma_M$ we will be able to provide an upper bound on (A.16). Our starting point is that $(\mathcal{S}, \mathcal{S}') \sim \mathscr{D}_{pq}^M \times \mathscr{D}_{pq}^M$ then for any $\sigma \in \Gamma_M$, the random variable $\sup_{f \in \mathcal{F}_{pq}} |\text{err}_{\mathcal{S}}^l[f] - \text{err}_{\mathcal{S}'}^l[f]|$ follows the same distribution as $\sup_{f \in \mathcal{F}_{pq}} |\text{err}_{\sigma(\mathcal{S})}^l[f] - \text{err}_{\sigma(\mathcal{S}')}^l[f]|$.

Therefore:

$$\begin{aligned}
&\mathbb{P}_{(\mathcal{S},\mathcal{S}')\sim\mathscr{D}_{pq}^{M}\times\mathscr{D}_{pq}^{M}}\left(\sup_{f\in\mathcal{F}_{pq}}|\mathrm{err}^{l}_{\mathcal{S}}[f]-\mathrm{err}^{l}_{\mathcal{S}'}[f]|\geq\frac{\epsilon}{2}\right)\\
&=\frac{1}{2^{M}}\sum_{\sigma\in\Gamma_{M}}\mathbb{P}_{\mathcal{S},\mathcal{S}'\sim\mathscr{D}_{pq}^{M}\times\mathscr{D}_{pq}^{M}}\left(\sup_{f\in\mathcal{F}_{pq}}|\mathrm{err}^{l}_{\sigma(\mathcal{S})}[f]-\mathrm{err}^{l}_{\sigma(\mathcal{S}')}[f]|\geq\frac{\epsilon}{2}\right)\\
&=\mathbb{E}_{(\mathcal{S},\mathcal{S}')\sim\mathscr{D}_{pq}^{M}\times\mathscr{D}_{pq}^{M}}\left[\frac{1}{2M}\sum_{\sigma\in\Gamma_{M}}\mathbb{I}\left[\sup_{f\in\mathcal{F}_{pq}}|\mathrm{err}^{l}_{\sigma(\mathcal{S})}[f]-\mathrm{err}^{l}_{\sigma(\mathcal{S}')}[f]|\geq\frac{\epsilon}{2}\right]\right]\\
&\leq\sup_{(\mathcal{S},\mathcal{S}')\in(\mathbb{X}\times\mathbb{Y})^{2M}}\left[\mathbb{P}_{\sigma\in\Gamma_{M}}\left(\sup_{f\in\mathcal{F}_{pq}}|\mathrm{err}^{l}_{\sigma(\mathcal{S})}[f]-\mathrm{err}^{l}_{\sigma(\mathcal{S}')}[f]|\geq\frac{\epsilon}{2}\right)\right] \qquad \text{(A.17)}
\end{aligned}$$

which concludes the second step.

**Reduction to a finite class.** The idea is to reduce $\mathcal{F}_{pq}$ in (A.17) to a finite class of functions. For the sake of conciseness, we will not enter into the details of the theory of *covering numbers*. Please refer to the corresponding literature for further details (*e.g.* [Devroye et al., 1996]).

In the following, $\Gamma(\epsilon/8,\mathcal{F}_{pq},2M)$ will denote the *uniform* $\epsilon/8$ *covering number* of $\mathcal{F}_{pq}$ over a sample of size $2M$.

Let define $\mathcal{G}_{pq}\subset\mathcal{F}_{pq}$ such that $(l^{\mathcal{G}_{pq}})_{|(\mathcal{S},\mathcal{S}')}$ is an $\epsilon/8$-cover of $(l^{\mathcal{F}_{pq}})_{|(\mathcal{S},\mathcal{S}')}$. Thus, $|\mathcal{G}_{pq}|\leq\Gamma(\epsilon/8,l^{\mathcal{F}_{pq}},2M)<\infty$ Therefore, if $\exists f\in\mathcal{F}_{pq}$ such that $|\mathrm{err}^{l}_{\sigma(\mathcal{S})}[f]-\mathrm{err}^{l}_{\sigma(\mathcal{S}')}[f]|\geq\frac{\epsilon}{2}$ then, $\exists g\in\mathcal{G}_{pq}$ such that $|\mathrm{err}^{l}_{\sigma(\mathcal{S})}[g]-\mathrm{err}^{l}_{\sigma(\mathcal{S}')}[g]|\geq\frac{\epsilon}{4}$ and the following comes naturally

$$\begin{aligned}
&\mathbb{P}_{\sigma\in\Gamma_{M}}\left(\sup_{f\in\mathcal{F}_{pq}}|\mathrm{err}^{l}_{\sigma(\mathcal{S})}[f]-\mathrm{err}^{l}_{\sigma(\mathcal{S}')}[f]|\geq\frac{\epsilon}{2}\right)\\
&\leq\mathbb{P}_{\sigma\in\Gamma_{M}}\left(\max_{g\in\mathcal{G}_{pq}}|\mathrm{err}^{l}_{\sigma(\mathcal{S})}[g]-\mathrm{err}^{l}_{\sigma(\mathcal{S}')}[g]|\geq\frac{\epsilon}{4}\right)\\
&\leq\Gamma(\epsilon/8,l^{\mathcal{F}_{pq}},2M)\max_{g\in\mathcal{G}_{pq}}\mathbb{P}_{\sigma\in\Gamma_{M}}\left(|\mathrm{err}^{l}_{\sigma(\mathcal{S})}[g]-\mathrm{err}^{l}_{\sigma(\mathcal{S}')}[g]|\geq\frac{\epsilon}{8}\right)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{(union bound)}
\end{aligned}$$

**Hoeffding's inequality.** Finally, consider $|\mathrm{err}^{l}_{\sigma(\mathcal{S})}[g]-\mathrm{err}^{l}_{\sigma(\mathcal{S}')}[g]|$ as the average of $M$ realizations of the same random variable, with expectation equal to 0. Then by Hoeffding's inequality we have that[1]

$$\mathbb{P}_{\sigma\in\Gamma_{M}}\left(|\mathrm{err}^{l}_{\sigma(\mathcal{S})}[g]-\mathrm{err}^{l}_{\sigma(\mathcal{S}')}[g]|\geq\frac{\epsilon}{4}\right)\leq 2e^{-M\epsilon^{2}/128} \qquad \text{(A.18)}$$

Putting everything together yields the result w.r.t. $\Gamma(\epsilon/8,l^{\mathcal{F}_{pq}},2M)$ for $M\epsilon^{2}\geq 32$. For $M\epsilon^{2}<32$ it holds trivially.

Recall that $l^{\mathcal{F}_{pq}}$ is Lipschitz in its first argument with a Lipschitz constant $L=1$ thus $\Gamma(\epsilon/8,l^{\mathcal{F}_{pq}},2M)\leq\Gamma(\epsilon/8,\mathcal{F}_{pq},2M)=\mathcal{O}\left(\left(\frac{8}{\epsilon}\right)^{\mathrm{Pdim}(\mathcal{F}_{pq})}\right)$ □

[1]Note that in some references the right-hand side of (A.18) might viewed as a probability measure over $M$ independent Rademacher variables.

The last part of the proof comes from the observation that, for any fixed $(p, q)$, we had never used any other specific information about $\mathcal{F}_{pq}$ other than the upper bound of $D+1$ over its pseudo dimension. In other words, equation (A.15) holds for slightly modified definition of $\mathcal{F}_{pq}$ as long as the pseudo dimension does not exceed $D+1$.

Let us now consider :

$$\widetilde{\mathcal{F}}_{pq} \doteq \{f : f(\boldsymbol{x}) \doteq \mathbb{I}\left[t(\boldsymbol{x}) = q\right] \mathbb{I}\left[\boldsymbol{x} \in \mathcal{A}_p^{\alpha}\right] \left\langle \mathbf{W}_{\cdot p} - \mathbf{W}_{\cdot q}; \boldsymbol{x} \right\rangle : \boldsymbol{w}_p, \boldsymbol{w}_q \in \mathcal{B}_1\}$$

Clearly for each function in $\widetilde{\mathcal{F}}_{pq}$ there is at most one corresponding affine function, thus $\widetilde{\mathcal{F}}_{pq}$ and $\mathcal{F}_{pq}$ share the same upper bound of $D+1$ on their pseudo-dimension.

Consequently, any covering number of $\mathcal{F}_{pq}$ is also a covering number of $\widetilde{\mathcal{F}}pq$. More precisely, this proof holds true for any $\boldsymbol{w}_p$ and $\boldsymbol{w}_q$, independently of $\mathcal{A}p^{\alpha}$ which may itself be defined with respect to $\boldsymbol{w}_p$ and $\boldsymbol{w}_q$.

It comes naturally that, fixing $\mathcal{S}$ as the training set, the following holds true:

$$\frac{1}{m}\sum_{M} \mathbb{I}\left[t(\boldsymbol{x}) = q\right] \mathbb{I}\left[\boldsymbol{x} \in \mathcal{A}_p^{\alpha}\right] \boldsymbol{x} = \boldsymbol{x}_{\text{up}}^{pq}$$

Thus

$$\left|\text{err}_{\mathcal{S}}^{l}[f] - \text{err}_{\mathscr{D}}^{l}[f]\right| = \left|\left\langle \frac{\boldsymbol{w}_p - \boldsymbol{w}_q}{\|\boldsymbol{w}_p - \boldsymbol{w}_q\|}; \boldsymbol{x}_{\text{up}}^{pq} \right\rangle - \left\langle \frac{\boldsymbol{w}_p - \boldsymbol{w}_q}{\|\boldsymbol{w}_p - \boldsymbol{w}_q\|}; \mu_q^p \right\rangle\right|$$

We can generalize this result for any couple $(p, q)$ by a simple union bound, giving the desired inequality:

$$\mathbb{P}_{(\mathbb{X}\times\mathbb{Y})\sim\mathscr{D}}\left(\sup_{\mathbf{W}\in\mathbb{R}^{D\times Q}} \left|\left\langle \frac{\boldsymbol{w}_p - \boldsymbol{w}_q}{\|\boldsymbol{w}_p - \boldsymbol{w}_q\|}; \boldsymbol{x}_{\text{up}}^{pq} \right\rangle - \left\langle \frac{\boldsymbol{w}_p - \boldsymbol{w}_q}{\|\boldsymbol{w}_p - \boldsymbol{w}_q\|}; \mu_q^p \right\rangle\right| \geq \epsilon\right) \leq \mathcal{O}\left(Q^2 \left(\frac{8}{\epsilon}\right)^{(D+1)} e^{M\epsilon^2/128}\right)$$

Equivalently, we have that

$$\left|\left\langle \frac{\boldsymbol{w}_p - \boldsymbol{w}_q}{\|\boldsymbol{w}_p - \boldsymbol{w}_q\|}; \boldsymbol{x}_{\text{up}}^{pq} \right\rangle - \left\langle \frac{\boldsymbol{w}_p - \boldsymbol{w}_q}{\|\boldsymbol{w}_p - \boldsymbol{w}_q\|}; \mu_q^p \right\rangle\right| \geq \epsilon$$

with probability $1-\delta$ for

$$M \in \mathcal{O}\left(\frac{1}{\epsilon^2}\left[\ln\left(\frac{1}{\delta}\right) + \ln(Q) + D\ln\left(\frac{1}{\epsilon}\right)\right]\right)$$

□

## A.3 Appendix of Part III

This section is divided in three. We first give an overview of the basic notions and results for the proofs to come, additionally, we will introduce some complementary notations in this introductory part. The two other sections consist in a rewriting of the proof of [Grunbaum, 1960] on the partition of convex bodies by hyperplanes. The proof is restated in full with proper notation and is the starting point of our result. The last section gives the proof of theorem 5.2 which is an extended version of the result of Grunbaum to approximate center of gravity.

### A.3.1 Preliminaries

#### Hyper-Sphere and Hyper-Ball

**Definition A.1** ($D$-dimensional Sphere) *We call n-sphere of center $\boldsymbol{o} \in \mathbb{R}^D$ and radius $R \in \mathbb{R}$ and write $\mathcal{SP}(\boldsymbol{o}, R) \subset \mathbb{R}^D$ the subset*

$$\mathcal{SP}(\boldsymbol{o}, R) \doteq \{\boldsymbol{x} \in \mathbb{R}^D : \|\boldsymbol{x} - \boldsymbol{o}\| = R\}$$

**Definition A.2** ($D$-dimensional Ball) *We call D-ball of center $\boldsymbol{o} \in \mathbb{R}^D$ and radius $R \in \mathbb{R}$ and write $\mathcal{B}(O, R) \subset \mathbb{R}^D$ the subset*

$$\mathcal{B}(\boldsymbol{o}, R) \doteq \{\boldsymbol{x} \in \mathbb{R}^D : \|\boldsymbol{x} - \boldsymbol{o}\| < R\}$$

*Alternatively, one can think of a ball as :*

$$\mathcal{B}(\boldsymbol{o}, R) \doteq \bigcup_{r \in [0,R]} \mathcal{SP}(\boldsymbol{o}, r)$$

**Definition A.3** (Surface of a sphere) *We call* Surface *of the D-sphere $\mathcal{SP}(\boldsymbol{o}, R)$ and write $Vol(\mathcal{SP}(\boldsymbol{o}, R))$ the $D-1$ dimensional volume*

$$Vol(\mathcal{SP}(\boldsymbol{o}, R)) \doteq \Pi_D^s \times R^{D-1}$$

*Where $\Pi_D^s$ is a constant factor depending only on $D$ (e.g $\Pi_1^s = 2$, $\Pi_2^s = 2\pi$, $\Pi_3^s = 4\pi$ and so on ...)*

**Definition A.4** (Volume of a Ball) *We call* Volume *of the D-ball $\mathcal{B}(\boldsymbol{o}, R)$ and write $Vol(\mathcal{B}(\boldsymbol{o}, R))$ the D-dimensional volume*

$$Vol(\mathcal{B}(\boldsymbol{o}, R)) \doteq \int_0^R Vol(\mathcal{SP}(\boldsymbol{o}, r))dr$$

*That is*

$$\begin{aligned} Vol(\mathcal{B}(\boldsymbol{o}, R)) &= \int_0^R \Pi_D^s R^{D-1} dr \\ &= \frac{\Pi_D^s R^D}{D} \\ &= \Pi_D R^D \end{aligned}$$

*Where $\Pi_D \doteq \frac{\Pi_D^s}{D}$ is a constant factor depending only on $D$.*

#### Hyper-Cone

From these core definitions, we can now introduce (Hyper)-cones and some of their core properties. Intuitively, an Hyper-cone of dimension $D+1$, center $\boldsymbol{o}$, radius $R$ and height $H$ is a sequence of $D$-Ball of linearly decreasing radius between $R$ and 0, each one living on a difference "slice" of $\mathbb{R}^{D+1}$ in such a way that the first Ball has center $\boldsymbol{o}$ and the last one $\boldsymbol{o} + H\boldsymbol{u}D + 1$.

**Remark A.2** *We will use $\boldsymbol{u}_{D+1}$ to denote the vector of $\mathbb{R}^{D+1}$ with 1 on its $D+1$ component and 0 elsewhere. Moreover, for the sake of notation conciseness, we will assume, without loss of generality, that hyper-cones are always aligned with the $D+1$ dimension of $\mathbb{R}^{D+1}$*

**Definition A.5** (Hyper-cone) *We call Hyper-cone of dimension $D+1$, base $\mathcal{B}(\boldsymbol{o}, R) \subset \mathbb{R}^D$ and height $H$ the set :*

$$H \doteq \bigcup_{\forall \eta \in [0,H]} \mathcal{B}\left(\boldsymbol{o} + \eta \boldsymbol{u}_{D+1}, \frac{H-\eta}{H} R\right)$$

Alternatively, we can define the *apex* $\boldsymbol{z} \doteq \boldsymbol{o} + H \times \boldsymbol{u}_{D+1}$ of the hyper-cone and give the following definition :

**Definition A.6** (Hyper-cone (2)) *We call Hyper-cone of dimension $D+1$, base $\mathcal{B}(\boldsymbol{o}, R) \subset \mathbb{R}^D$ and apex $\boldsymbol{z}$ the convex hull $\boldsymbol{conv}(\{\mathcal{B}(\boldsymbol{o}, R); \boldsymbol{z}\})$.*

We are now ready to state the core properties of Hyper-cone that we will use in the remaining of this document.

We start with the volume of a Hyper-cone

**Definition A.7** (Volume of Hyper-cone) *Given an Hyper-cone $\mathcal{C} \in \mathbb{R}^{D+1}$ of dimension $D+1$, base $\mathcal{B}(\boldsymbol{o}, R) \subset \mathbb{R}^D$ and height $H$ we call* volume *and write $Vol(\mathcal{C})$ the $D+1$-dimensional volume :*

$$Vol(\mathcal{C}) \doteq \int_0^H Vol\left(\mathcal{B}\left(\boldsymbol{o} + \eta \boldsymbol{u}_{D+1}, \frac{H-\eta}{H} R\right)\right) d\eta$$

**Proposition A.2** *The volume of the Hyper-cone $\mathcal{C} \subset \mathbb{R}^{D+1}$ of dimension $D+1$, base $\mathcal{B}(\boldsymbol{o}, R) \subset \mathbb{R}^D$ and height $H$ is*

$$Vol(\mathcal{C}) = \frac{\Pi_D R^D}{D+1} H$$

*Proof.* From the definition of volume of a sphere we have $\mathrm{Vol}(\mathcal{B}(\boldsymbol{o}, R)) \doteq \Pi_D R^D$. Substituting $R$ by $\frac{H-\eta}{R}$ and from the definition of the volume of a Hyper-cone we have

$$\mathrm{Vol}(\mathcal{C}) = \int_0^H \Pi_D \left(\frac{H-\eta}{H} R\right)^D d\eta$$

We substitute $\eta$ by $v \doteq H - \eta$, $dv = -d\eta$.

$$\begin{aligned}
\mathrm{Vol}(\mathcal{C}) &= \int_H^0 -\Pi_D \left(\frac{v}{H} R\right)^D dv \\
&= \int_0^H \Pi_D \left(\frac{v}{H} R\right)^D dv \\
&= \frac{\Pi_D R^D}{H^D} \int_0^H v^D dv \\
&= \frac{\Pi_D R^D}{H^D} \times \frac{H^{D+1}}{D+1} \\
&= \frac{\Pi_D R^D}{D+1} H
\end{aligned}$$

□

**Center of Mass**

We can now move on to the center of gravity. For a reminder of the definition of the center of gravity, we refer the reader to Definition 5.1

**Proposition A.3** *Let $\mathcal{SP}$ a D-dimensional sphere such that $\mathcal{SP} \doteq \mathcal{SP}(\boldsymbol{o}, R)$. Then $CG(\mathcal{SP}) = \boldsymbol{o}$.*

*Proof.* Without loss of generality, assume that $\boldsymbol{o} = 0$. Then, $\mathcal{SP} = \{\boldsymbol{x} \in \mathbb{R}^D : \|\boldsymbol{x}\| = R\}$. Since $\|\boldsymbol{x}\| = \| - \boldsymbol{x}\|$ it is clear that $\forall \boldsymbol{x} \in \mathbb{R}^D : \boldsymbol{x} \in \mathcal{SP} \Leftrightarrow -\boldsymbol{x} \in \mathcal{SP}$. Thus, we can rewrite $CG(\mathcal{SP})$ as

$$CG(\mathcal{SP}) = \frac{1}{\text{Vol}(\mathcal{SP})} \int_{x \in \mathcal{SP}} -\boldsymbol{x} d\boldsymbol{x}$$

Thus

$$\begin{aligned} 2CG(\mathcal{SP}) &= \frac{1}{\text{Vol}(\mathcal{SP})} \left( \int_{x \in \mathcal{SP}} x d\boldsymbol{x} + \int_{x \in \mathcal{SP}} -\boldsymbol{x} d\boldsymbol{x} \right) \\ &= \frac{1}{\text{Vol}(\mathcal{SP})} \int_{\boldsymbol{x} \in \mathcal{SP}} \boldsymbol{x} - \boldsymbol{x} d\boldsymbol{x} \\ &= 0 \end{aligned}$$

□

**Proposition A.4** *Let $\mathcal{B}$ a D-dimensional ball such that $\mathcal{B} \doteq \mathcal{B}(\boldsymbol{o}, R)$. Then $CG(\mathcal{B}) = \boldsymbol{o}$.*

*Proof.* Remind that $\mathcal{B}$ can be seen as a collection of concentric $D$-sphere of center $\boldsymbol{o}$ and radii between 0 and $R$ (see Def. A.2). Then, we can rewrite $CG(\mathcal{B})$ as

$$\begin{aligned} CG(\mathcal{B}) &= \frac{1}{\text{Vol}(\mathcal{B})} \int_0^R CG(\mathcal{SP}(\boldsymbol{o}, r)) \text{Vol}(\mathcal{SP}(\boldsymbol{o}, r)) dr \\ &= O \end{aligned}$$

Where the last line come from Proposition A.3. □

**Proposition A.5** (Center of Gravity of an Hyper-cone) *Let $\mathcal{C}$ a $D+1$ dimensional Hyper-cone ($\mathcal{C} \subset \mathbb{R}^{D+1}$) of base $\mathcal{B}(\boldsymbol{o}, R) \subset \mathbb{R}^D$ and apex $\boldsymbol{z}$ such that $\|\boldsymbol{z} - \boldsymbol{o}\| = H$. Then, $CG(\mathcal{C})$ is located on the segment $[O; \boldsymbol{z}]$ at a distance $H/_{D+2}$ of $\boldsymbol{o}$.*

*Proof.* Without loss of generality, we assume that $\boldsymbol{o} = 0$ and $\boldsymbol{z} = H\boldsymbol{u}_{D+1}$. By definition, $\mathcal{C}$ is a collection of ball, and we can rewrite $CG(\mathcal{C})$ as :

$$CG(\mathcal{C}) = \frac{1}{\text{Vol}(\mathcal{C})} \int_0^H CG\left[\mathcal{B}\left(\eta \boldsymbol{u}_{D+1}, \frac{H-\eta}{H} R\right)\right] \times \text{Vol}\left[\mathcal{B}\left(\eta \boldsymbol{u}_{D+1}, \frac{H-\eta}{H} R\right)\right] d\eta$$

From Proposition A.4 it is clear that $CG(\mathcal{C})$ lies on the segment $[\boldsymbol{o}; \boldsymbol{z}]$. The remaining of the proof came by explicitly calculating $CG(\mathcal{C})$.

$$
\begin{aligned}
CG(\mathcal{C}) &= \frac{1}{\text{Vol}(\mathcal{C})} \int_0^H \eta \boldsymbol{u}_{D+1} \times \text{Vol}\left[\mathcal{B}\left(\eta \boldsymbol{u}_{D+1}, \frac{H-\eta}{H} R\right)\right] d\eta \quad \text{(Prop. A.4)} \\
&= \frac{1}{\text{Vol}(\mathcal{C})} \int_0^H \eta \boldsymbol{u}_{D+1} \frac{\Pi_D R^D}{H^D} (H-\eta)^D d\eta \qquad \text{(Volume of a Ball)} \\
&= \frac{1}{\text{Vol}(\mathcal{C})} \int_0^H (H-v) \boldsymbol{u}_{D+1} \frac{\Pi_D R^D}{H^D} v^D dv \\
& \qquad\qquad \text{(Subst. } v \doteq H-\eta \text{, see Prop. A.2)} \\
&= \frac{1}{\text{Vol}(\mathcal{C})} \left[\int_0^H \frac{\Pi_D R^D H v_{D+1}}{H^D} v^D dv - \int_0^H \frac{\Pi_D R^D v_{D+1}}{H^D} v^{D+1} dv\right] \\
&= \frac{1}{\text{Vol}(\mathcal{C})} \left[\frac{\Pi_D R^D \boldsymbol{u}_{D+1}}{H^{D-1}} \int_0^H v^D dv - \frac{\Pi_D R^D \boldsymbol{u}_{D+1}}{H^D} \int_0^H v^{D+1} dv\right] \\
&= \frac{1}{\text{Vol}(\mathcal{C})} \left[\frac{\Pi_D R^D H^{D+1} \boldsymbol{u}_{D+1}}{H^{D-1}(D+1)} - \frac{\Pi_D R^D H^{D+2} \boldsymbol{u}_{D+1}}{H^D(D+2)}\right] \\
&= \frac{1}{\text{Vol}(\mathcal{C})} \left[\frac{\Pi_D R^D H^2 \boldsymbol{u}_{D+1}}{D+1} - \frac{\Pi_D R^D H^2 \boldsymbol{u}_{D+1}}{D+2}\right] \\
&= \frac{D+1}{\Pi_D R^D H} \left[\frac{\Pi_D R^D H^2 \boldsymbol{u}_{D+1}}{D+1} - \frac{\Pi_D R^D H^2 \boldsymbol{u}_{D+1}}{D+2}\right] \\
& \qquad\qquad \text{(Volume of a Hyper-cone)} \\
&= \left(H - H\frac{D+1}{D+2}\right) \boldsymbol{u}_{D+1} \\
&= \left(1 - \frac{D+1}{D+2}\right) H \boldsymbol{u}_{D+1} \\
&= \left(\frac{D+2-D-1}{D+2} H \boldsymbol{u}_{D+1}\right) \\
&= \left(\frac{H}{n+2}\right) \boldsymbol{u}_{D+1}
\end{aligned}
$$

That is to say, $CG(\mathcal{C})$ is on the segment $[\boldsymbol{o}; \boldsymbol{z}]$ at a distance $H/_{D+2}$ of $\boldsymbol{o}$. □

**Hyperplane and Halfspace**

We re-introduce hyperplane in this section and states an interesting property of hyper-cone relative to hyperplane that will be fundamental to the coming proof. Contrary to the main document, we will introduce hyperplane here with an offset term. This is done in order to ease the flow of the remaining proof, nonetheless it is not strictly necessary and we remind that as discussed in Section 1.1.2 the two settings are equivalent. Practically though, stating the full proofs without the introduction of an offset term would mean spending a lot of time in adding additional dimensions and dealing with unnecessary basis change throughout the course of this appendix.

**Definition A.8** *We call* $(D)$-Hyperplane *of normal* $\boldsymbol{w} \in \mathbb{R}^D$ *and offset* $b \in \mathbb{R}$ *and write* $h(\boldsymbol{w}, b) \subset \mathbb{R}^D$ *the subset :*

$$h(\boldsymbol{w}, b) \doteq \{\boldsymbol{x} \in \mathbb{R}^D : \langle \boldsymbol{w}; \boldsymbol{x} \rangle + b = 0\}$$

**Definition A.9** *We call* Positive Halfspace *of the D-Hyperplane* $h(\boldsymbol{w}, b) \subset \mathbb{R}^D$ *and write* $h^+(\boldsymbol{w}, b) \subset \mathbb{R}^D$ *the subset*

$$h^+(\boldsymbol{w}, b) \doteq \{\boldsymbol{x} \in \mathbb{R}^D : \langle \boldsymbol{w}; \boldsymbol{x} \rangle + b \geq 0\}$$

*Conversely, we call* Negative Halfspace *of* $h(\boldsymbol{w}, b) \subset \mathbb{R}^D$ *and write* $h^-(\boldsymbol{w}, b) \subset \mathbb{R}^D$ *the subset*

$$h^-(\boldsymbol{w}, b) \doteq \{\boldsymbol{x} \in \mathbb{R}^D : \langle \boldsymbol{w}; \boldsymbol{x} \rangle + b \leq 0\}$$

Additionally, note that $h(\boldsymbol{w}, b) \subset h^+(\boldsymbol{w}, b)$ but $h(\boldsymbol{w}, b) \not\subset h^-(\boldsymbol{w}, b)$.

**Definition A.10** *For any subset* $\mathcal{E} \subset \mathbb{R}^D$ *and any Hyperplane* $h \subset \mathbb{R}^D$ *we call* Positive Partition *and write* $\mathcal{E}^+ \subset \mathbb{R}^D$ *the subset*

$$\mathcal{E}^+ \doteq \mathcal{E} \cap h^+$$

*Conversely, for we call* Negative Partition *and write* $\mathcal{E}^- \subset \mathbb{R}^D$ *the subset*

$$\mathcal{E}^- \doteq \mathcal{E} \cap h^-$$

**Proposition A.6** (Volume reduction of Hyper-Cone) *For any* $(D+1)$*-Hyper-cone of base* $\mathcal{B}(\boldsymbol{o}, R)$*, apex* $\boldsymbol{z}$ *and Height* $H$*, let set* $h_{CG(\mathcal{C})} \doteq h(\boldsymbol{u}_{D+1}, H/_{n+2})$ *the Hyperplane passing by* $CG(\mathcal{C})$ *( i.e.* $CG(\mathcal{C}) \in h_{CG(\mathcal{C})}$ *) and parallel to* $\mathbb{R}^D$*. Then,*

$$Vol(\mathcal{C}^+) = Vol(\mathcal{C}) \left[ \frac{1}{\left(1 + \frac{1}{D+1}\right)^{D+1}} \right] \geq Vol(\mathcal{C}) e^{-1}$$

*Proof.* We start by proving the right-hand side of the relation. Let set $D' = D + 1$ and divide both side by $\mathrm{Vol}(\mathcal{C})$ then we can rewrite it as

$$\frac{1}{\left(1 + \frac{1}{D'}\right)^{D'}} \geq e^{-1}$$

From the usual definition of $e$ we have that

$$\lim_{D \to \infty} \left(1 + \frac{1}{D'}\right)^{D'} = e$$
$$\Leftrightarrow \lim_{D' \to \infty} \frac{1}{\left(1 + \frac{1}{D'}\right)^{D'}} = e^{-1}$$

And by standard arguments we can show that

$$\frac{1}{\left(1 + \frac{1}{D'}\right)^{D'}} \geq \frac{1}{\left(1 + \frac{1}{D'+1}\right)^{D'+1}}$$

Therefore

$$\frac{1}{\left(1 + \frac{1}{D'}\right)^{D'}} \geq \lim_{n \to \infty} \frac{1}{\left(1 + \frac{1}{D'}\right)^{D'}} = e^{-1}$$

Finally, the left-hand side of relation is obtained by direct calculation of $\mathrm{Vol}(\mathcal{C}^+)$. The general idea is the same than the calculation of $\mathrm{Vol}(\mathcal{C})$ but, instead of integrating over the entire height we start at $H/_{D+2}$, thus

ignoring $C^-$. Besides, without loss of generality, we assume that $\boldsymbol{o} = 0$ and that $\boldsymbol{z} = H\boldsymbol{u}_{D+1}$.

$$
\begin{aligned}
\mathrm{Vol}(\mathcal{C}^+) &= \int_{H/_{D+2}}^{H} \mathrm{Vol}\left[\mathcal{B}\left(\eta \boldsymbol{u}_{D+1}, \frac{H-\eta}{H} R\right)\right] d\eta \\
&= \int_{H/_{D+2}}^{H} \Pi_D \frac{R^D}{H^D} (H-\eta)^D d\eta \qquad \text{(Volume of a Ball)} \\
&= \int_{0}^{H\left(1-\frac{1}{D+2}\right)} \Pi_D \frac{R^D}{H^D} v^D dv \qquad \text{(Subst. } v \doteq H-\eta\text{)} \\
&= \frac{\Pi_D R^D}{H^D} \int_{0}^{H\left(1-\frac{1}{D+2}\right)} v^D dv \\
&= \frac{\Pi_D R^D}{H^D} \times \frac{H^{D+1}}{D+1} \times \left(1-\frac{1}{D+2}\right)^{D+1} \\
&= \frac{\Pi_D R^D H}{D+1} \times \left(1-\frac{1}{D+2}\right)^{D+1} \\
&= \mathrm{Vol}(\mathcal{C}) \left(1-\frac{1}{D+2}\right)^{D+1} \\
&= \mathrm{Vol}(\mathcal{C}) \left(\frac{D+1}{D+2}\right)^{D+1} \\
&= \mathrm{Vol}(\mathcal{C}) \left(\frac{(D+1)\times\frac{1}{D+1}}{(D+2)\times\frac{1}{D+1}}\right)^{D+1} \\
&= \mathrm{Vol}(\mathcal{C}) \left(\frac{1}{\frac{D+2}{D+1}}\right)^{D+1} \\
&= \mathrm{Vol}(\mathcal{C}) \left(\frac{1}{1+\frac{1}{D+1}}\right)^{D+1} \\
&= \mathrm{Vol}(\mathcal{C}) \left[\frac{1}{\left(1+\frac{1}{D+1}\right)^{D+1}}\right]
\end{aligned}
$$

□

**Spherical Symmetric**

For the remaining of this document, let $\mathcal{K}$ be a (full dimensional) convex body in $\mathbb{R}^{D+1}$.

**Definition A.11** (Spherical Symmetric) *For any convex body $K \in \mathbb{R}^{D+1}$ we say that $\mathcal{K}$ is* Spherically Symmetric along the unit vector $\boldsymbol{v}$ *if and only if $\forall b \in \mathbb{R}$ the cut of $\mathcal{K}$ by the hyperplane $h(\boldsymbol{v}, b)$ (i.e. $\mathcal{K} \cap h(\boldsymbol{v}, b)$) is a $D$ dimensional hypersphere of center $b\boldsymbol{v}$*

### A.3.2 Partition of Convex Bodies By Hyper-Plane

This section consists in a rewriting of the proof of [Grunbaum, 1960] instantiated within the previously defined notations and setting.

We first recall the theorem for easy reference:

**Theorem A.2** *For any convex body $\mathcal{K} \subset \mathbb{R}^{D+1}$, and any hyperplane $h$. If $CG(\mathcal{K}) \in \mathcal{K}^+$ then*

$$Vol(\mathcal{K}^+) \geq e^{-1} \times Vol(\mathcal{K})$$

*Proof of Theorem 5.1.*

**Remark A.3** (Points along $\boldsymbol{u}_{D+1}$) *This proof will revolve around key points located on the $D+1^{th}$ axis of $\mathbb{R}^{D+1}$ of base vector $\boldsymbol{u}_{D+1}$. In an attempt to avoid overburdening the notation, we will treat these points as number along the real line when context is clear. Therefore, if $\boldsymbol{x}_1 = \lambda_1 \boldsymbol{u}_{D+1}$ and $\boldsymbol{x}_2 = \lambda_2 \boldsymbol{u}_{D+1}$ we will freely write $\boldsymbol{x}_1 > \boldsymbol{x}_2'$ if $\lambda_1 > \lambda_2$.*

Let $h$ the hyperplane such that $h = \arg\min_h \text{Vol}(\mathcal{K}^+)$ such that $CG(\mathcal{K}) \in \mathcal{K}^+$. It is easy to see that $CG(\mathcal{K}) \in h$: if $CG(\mathcal{K}) \notin h$ you can always reduce $\text{Vol}(\mathcal{K}^+)$ by shifting $h$ toward $CG(\mathcal{K})$. Without loss of generality, let's say that $CG(\mathcal{K}) = \mathbf{0}$ the origin of $\mathbb{R}^{D+1}$ and that $\boldsymbol{u}_{D+1}$ is the normal vector of $h$ with $\lambda = 0$, hence $h = h(\boldsymbol{u}_{D+1}, 0)$.

In order to ease the comprehension of the proof, we make the following assumption that we will lift later on.

**Assumption A.1** *$\mathcal{K}$ is a convex body which is Spherically Symmetric along $\boldsymbol{u}_{D+1}$*

A direct implication of this is that $CG(\mathcal{K}) = CG(\mathcal{K} \cap h)$. In other words, $CG(\mathcal{K})$ is the center of the $n-1$ dimensional sphere $\mathcal{K} \cap h$ (see, for example, the argument of proposition. A.5 ).

Let $\mathcal{C}^+$ the Hyper-cone of base $\mathcal{K} \cap h$ and apex $\boldsymbol{z}$ such that $\mathcal{C}^+ \subset h^+$ and $\text{Vol}(\mathcal{C}^+) = \text{Vol}(\mathcal{K}^+)$.

Moreover either :

- $\mathcal{K}^+ = \mathcal{C}^+$ and $\boldsymbol{z}$ is the apex of $\mathcal{K}^+$
- $\boldsymbol{z} \notin \mathcal{K}^+$

To prove that, remember that each *slice* $\mathcal{K} \cap h(\boldsymbol{u}_{D+1}, \lambda)$ of $\mathcal{K}$ along the $D+1$ axis is a sphere. We look at the function $f_{\mathcal{C}^+}()$ (resp. $f_{\mathcal{K}^+}()$) which maps each value of $\lambda \in \mathbb{R}_+$ with the radius of the corresponding *slice* of $\mathcal{C}^+$ (resp. $\mathcal{K}^+$).By construction, we know that $f_{\mathcal{C}^+}(0) = f_{\mathcal{K}^+}(0)$ and by definition $f_{\mathcal{C}^+}()$ is a decreasing linear function. If $f_{\mathcal{K}^+}()$ has any strictly convex part, then there exists an arc $[f_{\mathcal{K}^+}(\lambda_1), r_{\mathcal{K}^+}(\lambda_2)]$ which is not in $\mathcal{K}^+$ and therefore $\mathcal{K}$ is not a convex set. Hence $f_{K^+}()$ is concave. Then, because $f_{\mathcal{C}^+}(0) = f_{\mathcal{K}^+}(0)$, for $\boldsymbol{z}$ to be in $\mathcal{K}^+$ either $\mathcal{K}^+$ is a Hyper-cone (and $f_{\mathcal{K}^+}$ is linear) or $\text{Vol}(\mathcal{K}^+) > \text{Vol}(\mathcal{C}^+)$ (that is to say $\int_0^\infty f_{\mathcal{K}^+}(\lambda) d\lambda > \int_0^\infty f_{\mathcal{C}^+}(\lambda) d\lambda$ ) which is in contradiction with the definition of $\mathcal{C}^+$

As a consequence, $\mathcal{C}^+$ is at least as elongated as $\mathcal{K}^+$. In other words, the mass of $\mathcal{C}^+$ is more spread along the axis of $\boldsymbol{u}_{D+1}$, this incurs a shift of the center of gravity of $CG(\mathcal{C}^+)$ with respect to $CG(\mathcal{K}^+)$. Therefore $CG(\mathcal{C}^+)$ is on the closed segment $[CG(\mathcal{K}^+), \boldsymbol{z}]$.

Thus, by using the notation introduced in Note A.3 :

$$0 = CG(\mathcal{K}) \leq CG(\mathcal{K}^+) \leq CG(\mathcal{C}^+) \leq \boldsymbol{z}$$

We now define $\mathcal{C}^-$ by extending $\mathcal{C}^+$ such that $\mathcal{C} \doteq \mathcal{C}^- \cup \mathcal{C}^+$ is a cone of

apex $z$ and $\text{Vol}(\mathcal{C}^-) = \text{Vol}(\mathcal{K}^-)$. Therefore,

$$\begin{aligned}\text{Vol}(\mathcal{C}) &= \text{Vol}(\mathcal{C}^+) + \text{Vol}(\mathcal{C}^-) \\ &= \text{Vol}(\mathcal{K}^+) + \text{Vol}(K^-) \\ &= \text{Vol}(\mathcal{K})\end{aligned}$$

Once again, we are interested in the relative position of $CG(\mathcal{K}^-)$ and $CG(\mathcal{C}^-)$. We invoke the same arguments than before and claim that, in a similar way :

$$CG(\mathcal{K}^-) \leq CG(C^-) \leq 0 = CG(\mathcal{C})$$

**Remark A.4** *The proof for this is a little more tricky this time though. Part of this is due to the fact that $\mathcal{C}^-$ is not a hyper-cone in itself and one must consider $\mathcal{C}$ and $\mathcal{K}$ in their entirety for the non-convexity argument. A possible start is to consider the radius increase along the reverse axis $\overline{\boldsymbol{u}_{D+1}} \doteq -\boldsymbol{u}_{D+1}$ and replicate the previous argument with added attention to the slope of $f_{\mathcal{K}^-}()$ which must be such that $f_{\mathcal{K}}()$ as a whole is still concave.*

Let $\alpha, \beta \in \mathbb{R}$ such that $\alpha \doteq \text{Vol}(\mathcal{K}^+)/_{\text{Vol}(\mathcal{K})}$ and $\beta \doteq \text{Vol}(\mathcal{K}^-)/_{\text{Vol}(\mathcal{K})}$. Then

$$CG(\mathcal{K}) = \alpha CG(\mathcal{K}^+) + \beta CG(\mathcal{K}^-)$$

Or alternatively, by construction of $\mathcal{C}$

$$CG(\mathcal{C}) = \alpha CG(\mathcal{C}^+) + \beta CG(\mathcal{C}^-)$$

Combining these with the previous inequalities, we have that

$$CG(\mathcal{K}) \leq CG(\mathcal{C})$$

Moreover, we know from proposition A.5 that $CG(\mathcal{C})$ is at a distance $H/_{D+2}$ of its base, where $H$ is the height of $\mathcal{C}$.

Let $\widetilde{h} \doteq h(\boldsymbol{u}_{D+1}, \tilde{b})$ such that $CG(\mathcal{C}) \in \widetilde{h}$ and write $\widetilde{\mathcal{C}^+}$ the positive partition of $\mathcal{C}$ by $\widetilde{h}$, that is $\widetilde{\mathcal{C}^+} \doteq \widetilde{h^+} \cap \mathcal{C}$.

From Proposition A.6 we have that $\text{Vol}\left(\widetilde{\mathcal{C}^+}\right) \geq e^{-1}\text{Vol}(\mathcal{C})$. Moreover, because of $CG(\mathcal{C}) \geq CG(\mathcal{K})$ we have that $\text{Vol}(\mathcal{C}^+) \geq \text{Vol}\left(\widetilde{\mathcal{C}^+}\right)$.

Putting all of this together we get that

$$\begin{aligned}\text{Vol}(\mathcal{K}^+) &= \text{Vol}(\mathcal{C}^+) \\ &\geq \text{Vol}\left(\widetilde{\mathcal{C}^+}\right) \\ &\geq e^{-1}\text{Vol}(\mathcal{C}) \\ &= e^{-1}\text{Vol}(\mathcal{K})\end{aligned}$$

Finally, all we have left is to deal with assumption A.1. This is simply tackled by remarking that, by definition, spherical symmetrization preserve volumes for sliced part along the symmetry axis. Thus, for any $\mathcal{K}$ of any convex shape it suffices to apply the proof on the spherical symmetrization of $\mathcal{K}$ : $\mathbf{sym}_S(\mathcal{K})$ and we have

$$\text{Vol}(\mathcal{K}^+) = \text{Vol}(\mathbf{sym}_S(\mathcal{K}^+)) \geq e^{-1}\text{Vol}(\mathbf{sym}_S(\mathcal{K})) = \text{Vol}(\mathcal{K})$$

□

### A.3.3 Generalized Volume Reduction

This section is dedicated to theorem 5.2 which is a generalization of theorem 5.1 to approximate center of mass. As usual, we will first restate the theorem for clarity.

**Theorem A.3** *For any convex body $\mathcal{K} \subset \mathbb{R}^{D+1}$ and any hyperplane $h$ of normal $\boldsymbol{v}$, splitting $\mathcal{K}$ in $\mathcal{K}^+$ and $\mathcal{K}^-$. Let*

$$\boldsymbol{w} \doteq CG(\mathcal{K}) + \lambda \frac{(D+1)Vol(\mathcal{K})}{\Pi_D R_{\mathcal{K}^+}^D} \left[\frac{H_{\mathcal{K}^+}}{(D+2)H_{\mathcal{K}^-}}\right]^D \left[1 - \frac{1}{D+2}\right] \boldsymbol{v}$$

*Where $H_{\mathcal{K}^+} = \max_{\boldsymbol{x} \in \mathcal{K}^+} \boldsymbol{x}^\top \boldsymbol{v}$, $H_{\mathcal{K}^-} = \min_{\boldsymbol{x} \in \mathcal{K}^-} \boldsymbol{x}^\top \boldsymbol{v}$ and $R_{\mathcal{K}^+}$ the radius of the $D-1$-Ball $\mathcal{B}_{\mathcal{K} \cap h}$ such that $Vol(\mathcal{B}_{\mathcal{K}\cap h}) = Vol(\mathcal{K} \cap h)$.*

*Then, if $\boldsymbol{w} \in \mathcal{K}^+$ the following holds true*

$$Vol(\mathcal{K}^+) \geq Vol(\mathcal{K})(1-\lambda)^{D+1} e^{-1}$$

*Proof.* The proof start in a similar way than the one of Grunbaum, with respect to $\boldsymbol{w}$.

Namely, let assumption A.1 hold for now, and let $h$ the hyperplane such that $h = \arg\min_h \mathrm{Vol}(K^+)$ such that $\boldsymbol{w} \in \mathcal{K}^+$. Same as before, we have that $\boldsymbol{w} \in h$. Without loss of generality, let's say that $\boldsymbol{w} = 0$ the origin of $\mathbb{R}^{D+1}$ and that $\boldsymbol{u}_{D+1}$ is the normal vector of $h$ with $b = 0$, hence $h = h(\boldsymbol{u}_{D+1}, 0)$.

Let define $\mathcal{C}^+$ the Hyper-cone of base $\mathcal{K} \cap h$, apex $\boldsymbol{z}$ such that $\mathcal{C}^+ \subset h^+$ and $\mathrm{Vol}(\mathcal{C}^+) = \mathrm{Vol}(\mathcal{K}^+)$. Moreover, let $\mathcal{C}^-$ the extension of $\mathcal{C}^+$ such that $\mathcal{C} \doteq \mathcal{C}^- \cap \mathcal{C}^+$ is an hyper-cone of height $H$ and volume $\mathrm{Vol}(\mathcal{C}) = \mathrm{Vol}(\mathcal{K})$. That is to say $\mathrm{Vol}(\mathcal{C}^-) = \mathrm{Vol}(\mathcal{K}^-)$. From the same argument than before, we know that $CG(\mathcal{C}^+)$ (resp. $CG(\mathcal{C}^-)$) is shifted with respect to $CG(\mathcal{K}^+)$ (resp. $CG(\mathcal{K}^-)$), thus, according to the notation defined in note A.3 we have that

$$CG(\mathcal{K}) \leq CG(\mathcal{C})$$

If $\boldsymbol{w} \leq CG(\mathcal{C})$ then the exact same argument than the one of section A.3.2 applies and

$$\mathrm{Vol}(\mathcal{K}^+) \geq \mathrm{Vol}(\mathcal{K}) e^{-1} \qquad \text{(See Th. 5.1 for details)}$$

Otherwise, we have that

$$CG(\mathcal{K}) \leq CG(\mathcal{C}) \leq \boldsymbol{w}$$

The idea of the proof is to find $\widetilde{\boldsymbol{w}}$ such that $\boldsymbol{w} \leq \widetilde{\boldsymbol{w}}$ from which we can bound the volume of $\mathcal{K}^+$ in a similar way than before.

Let define

$$\widetilde{\boldsymbol{w}} \doteq CG(\mathcal{C}) + \lambda H \left[1 - \frac{1}{D+1}\right] \boldsymbol{u}_{D+1} \tag{A.19}$$

Denote by $\mathcal{B}_0 = \mathcal{B}(\boldsymbol{o}_0, R)$ the base of $\mathcal{C}$ and remind that $\mathcal{C}$ has height $H$, apex $\boldsymbol{z}$ and $CG(\mathcal{C}) = \boldsymbol{o}_0 + H/_{D+2}\boldsymbol{u}_{D+1}$. Therefore

$$\widetilde{\boldsymbol{w}} = H \left[\lambda \left(1 - \frac{1}{D+2}\right) + \frac{1}{D+2}\right] \boldsymbol{u}_{D+1}$$

Consider $\widetilde{h} \doteq h(\boldsymbol{u}_{D+1}, \widetilde{b})$ the hyperplane of normal vector $\boldsymbol{u}_{D+1}$ (i.e. $\widetilde{h}$ is parallel to $h$) and offset $\widetilde{b}$ such that $\widetilde{\boldsymbol{w}} \in \widetilde{h}$. Then, let $\widetilde{\mathcal{C}^+}$ the positive partition of $\mathcal{C}$ with respect to $\widetilde{h}$

$$\widetilde{\mathcal{C}^+} = \mathcal{C} \cap \widetilde{h^+}$$

We can compute the volume of $\widetilde{\mathcal{C}^+}$ as follow :

$$\begin{aligned}
\mathrm{Vol}(\widetilde{\mathcal{C}^+}) &= \int_{\widetilde{\boldsymbol{w}}}^{H} \mathrm{Vol}\left[\mathcal{B}\left(\boldsymbol{o}_0 + \eta \boldsymbol{u}_{D+1}, \frac{H-\eta}{H} R\right)\right] d\eta \\
&= \int_{\widetilde{\boldsymbol{w}}}^{H} \Pi_D \frac{R^D}{H^D} (H-\eta)^D d\eta && \text{(Volume of a Ball)} \\
&= \int_{0}^{H - H\left[\lambda\left(1-\frac{1}{D+2}\right)+\frac{1}{D+2}\right]} \Pi_D \frac{R^D}{H^D} v^D dv && \text{(Subst. } v \doteq H-\eta\text{)} \\
&= \frac{\Pi_D R^D}{H^D} \int_{0}^{H(1-\lambda)\left[1-\frac{1}{D+2}\right]} v^D dv \\
&= \frac{\Pi_D R^D}{H^D} \times \frac{H^{D+1}}{D+1} (1-\lambda)^{D+1} \left[1 - \frac{1}{D+2}\right]^{D+1} \\
&= \mathrm{Vol}(\mathcal{C})(1-\lambda)^{D+1} \left[1 - \frac{1}{D+2}\right]^{D+1} && \text{(Volume of a Hyper-cone)} \\
&\geq \mathrm{Vol}(\mathcal{C})(1-\lambda)^{D+1} e^{-1} && \text{(See Prop. A.6)}
\end{aligned}$$

Where in the first two lines, we allow a slight abuse of notation and use $\widetilde{\boldsymbol{w}}$ as a real as explained in Note A.3.

Then, we can rewrite the volume of $\mathcal{C}^+$ as

$$\mathrm{Vol}(\mathcal{C}^+) = \int_{\boldsymbol{w}}^{\widetilde{\boldsymbol{w}}} \mathrm{Vol}\left[\mathcal{B}\left(\boldsymbol{o}_0 + \eta \boldsymbol{u}_{D+1}, \frac{H-\eta}{H} R\right)\right] d\eta + \mathrm{Vol}(\widetilde{\mathcal{C}^+})$$

Consequently, if $\boldsymbol{w} \leq \widetilde{\boldsymbol{w}}$ then the first term of $\mathrm{Vol}(\mathcal{C}^+)$ is positive and therefore, $\mathrm{Vol}(\mathcal{C}^+) \geq \mathrm{Vol}(\widetilde{\mathcal{C}+})$

$$\begin{aligned}
\widetilde{\boldsymbol{w}} &= CG(\mathcal{C}) + \lambda H \left[1 - \frac{1}{D+2}\right] \\
&= CG(\mathcal{C}) + \lambda \frac{(D+1)\mathrm{Vol}(\mathcal{C})}{\Pi_D R^D} \left[1 - \frac{1}{D+2}\right] && \text{(Volume of a Hyper-cone)} \\
&\geq CG(\mathcal{K}) + \lambda \frac{(D+1)\mathrm{Vol}(\mathcal{K})}{\Pi_D R^D} \left[1 - \frac{1}{D+2}\right] \\
& && (CG(\mathcal{C}) \geq CG(\mathcal{K}) \text{ and } \mathrm{Vol}(\mathcal{C}) = \mathrm{Vol}(\mathcal{K}))
\end{aligned}$$

Unfortunately, we cannot easily compute $R$ directly. Nonetheless, since $\mathcal{B}_0$ and $h$ are parallel, we can use the triangle proportionality theorem. Denote $R_{\mathcal{C}^+}$ the radius of the base of $\mathcal{C}^+$, that is $\mathcal{K} \cap h$ and $H_{\mathcal{C}^+}$ the height of $\mathcal{C}^+$ (i.e. the distance between $\boldsymbol{w}$ and $\boldsymbol{z}$) then we have :

$$\frac{1}{R} = \frac{H_{\mathcal{C}^+}}{H R_{\mathcal{C}^+}}$$

From this, we want to bound $H_{\mathcal{C}^+}$ and $H$ since $R_{\mathcal{C}^+}$ is easy enough to estimate because it is directly related to $\mathcal{K}$ and $h$.

From previous argument, we know that $\boldsymbol{z} \notin \mathcal{K}^+$ except if $\mathcal{K}^+ = \mathcal{C}^+$. So let define

$$H_{\mathcal{K}^+} \doteq \max_{\boldsymbol{x} \in \mathcal{K}^+} \boldsymbol{x}^\top \boldsymbol{u}_{D+1}$$

Intuitively, $H_{\mathcal{K}^+}$ is maximal distance between a point in $\mathcal{K}^+$ and $\boldsymbol{w}$ with respect to the axis of $\boldsymbol{u}_{D+1}$. Because $\boldsymbol{z}$ is precisely on this axis, and that $\boldsymbol{z} \notin \mathcal{K}^+$ (or $\mathcal{K}^+ = \mathcal{C}^+$) the following holds true

$$H_{\mathcal{K}^+} \leq H_{\mathcal{C}^+} \tag{A.20}$$

Conversely, let define $H_{\mathcal{K}^-}$ as the maximal distance between $\boldsymbol{w}$ and any point of $\mathcal{K}^-$ with respect to the axis of $\boldsymbol{u}_{D+1}$. Again, from previous argument, we know that $\boldsymbol{o}_0 \in \mathcal{K}^-$. Note that, because $\mathcal{C}$ is a Hyper-cone, we know that $\boldsymbol{o}_0$ is at a distance $H/_{n+2}$ of $CG(\mathcal{C})$. Moreover, remind that we are treating the case where $\boldsymbol{w} \geq CG(\mathcal{C})$, hence, the distance between $\boldsymbol{o}_0$ and $\boldsymbol{w}$ is at least $H/_{D+2}$ which in turn is smaller than $H_{\mathcal{K}^-}$. Reordering gives the following

$$\begin{aligned} H_{\mathcal{K}^-} &\geq \frac{H}{D+2} \\ \Leftrightarrow (D+2) H_{\mathcal{K}^-} &\geq H \end{aligned} \tag{A.21}$$

Putting back equations (A.20) and (A.21) together, we have that

$$\frac{1}{R} \geq \frac{H_{\mathcal{K}^+}}{(D+2) H_{\mathcal{K}^-} R_{\mathcal{C}^+}}$$

Which we plug back into the previous calculation

$$\begin{aligned} \widetilde{\boldsymbol{w}} &\geq CG(\mathcal{K}) + \lambda \frac{(D+1)\mathrm{Vol}(\mathcal{K})}{\Pi_D R^D} \left[1 - \frac{1}{D+2}\right] \\ &\geq CG(\mathcal{K}) + \lambda \frac{(D+1)\mathrm{Vol}(\mathcal{K})}{\Pi_D R_{\mathcal{C}^+}^D} \left[\frac{H_{\mathcal{K}^+}}{(D+2) H_{\mathcal{K}^-}}\right]^D \left[1 - \frac{1}{D+2}\right] \\ &= \boldsymbol{w} \end{aligned}$$

Remind that we drop $\boldsymbol{u}_{D+1}$ in the above since we treat $\widetilde{\boldsymbol{w}}$ and $\boldsymbol{w}$ as real numbers (see Note A.3).

We conclude this proof by putting everything together. Namely,

- $\boldsymbol{w} \leq CG(\mathcal{C})$ and
$$\mathrm{Vol}(\mathcal{K}^+) \geq \mathrm{Vol}(\mathcal{K}) e^{-1}$$
- $\boldsymbol{w} \geq CG(\mathcal{C})$ and we can define $\widetilde{\boldsymbol{w}}$ such that
$$\mathrm{Vol}(\mathcal{K}^+) \geq \mathrm{Vol}(\widetilde{\mathcal{C}^+}) \geq \mathrm{Vol}(\mathcal{K})(1-\lambda)^D e^{-1}$$

Once again, we lift assumption A.1 as before by noting that spherical symmetry preserves volumes. One difference though lies in the fact that computing $R_{\mathcal{C}^+}$ is no longer immediate in the general case. Notwithstanding, it can be easily approximated within satisfactory precision.

As a final note, we may mention that distinguishing between these two cases is non-trivial. Hence, without additional computation, only the worst bound can be guaranteed. □

# Bibliography


[Amir and Mach, 1984] Amir, D. and Mach, J. (1984). Chebyshev centers in normed spaces. *Journal of approximation theory*, 40(4):364–374. (Cited in page 87.)

[Angluin and Laird, 1987] Angluin, D. and Laird, P. D. (1987). Learning from noisy examples. *Machine Learning*, 2(4):343–370. (Cited in page 38.)

[Bach and Jordan, 2002] Bach, F. R. and Jordan, M. I. (2002). Kernel Independent Component Analysis. *Journal of Machine Learning Research*, 3:1–48. (Cited in page 62.)

[Bache and Lichman, 2013] Bache, K. and Lichman, M. (2013). UCI machine learning repository. (Cited in page 66.)

[Balcan et al., 2007] Balcan, M., Broder, A. Z., and Zhang, T. (2007). Margin based active learning. In *Learning Theory, 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA, June 13-15, 2007, Proceedings*, pages 35–50. (Cited in page 125.)

[Bartlett et al., 1994] Bartlett, P. L., Long, P. M., and Williamson, R. C. (1994). Fat-shattering and the learnability of real-valued functions. In *Proceedings of the seventh annual conference on Computational learning theory*, pages 299–310. (Cited in page 19.)

[Ben-David et al., 1995] Ben-David, S., Cesa-Bianchi, N., Haussler, D., and Long, P. M. (1995). Characterizations of learnability for classes of {0, ..., n}-valued functions. *J. Comput. Syst. Sci.*, 50(1):74–86. (Cited in page 19.)

[Bennett and Demiriz, 1998] Bennett, K. P. and Demiriz, A. (1998). Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, pages 368–374. MIT Press. (Cited in page 65.)

[Bennett and Parrado-Hernández, 2006] Bennett, K. P. and Parrado-Hernández, E. (2006). The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7:1265–1281. (Cited in page 77.)

[Blanchard and Zwald, 2008] Blanchard, G. and Zwald, L. (2008). Finite-dimensional projection for classification and statistical learning. *IEEE Transactions on Information Theory*, 54(9):4169–4182. (Cited in page 62.)

[Block, 1962] Block, H.-D. (1962). The perceptron: A model for brain functioning. i. *Reviews of Modern Physics*, 34(1):123. (Cited in pages 20, 32, 55, and 112.)

[Blum et al., 1998] Blum, A., Frieze, A. M., Kannan, R., and Vempala, S. (1998). A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1/2):35–52. (Cited in pages ix, 4, 38, 39, 40, 41, 42, 60, 61, 133, 136, and 170.)

[Boser et al., 1992] Boser, B. E., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992*, pages 144–152. (Cited in pages 13 and 21.)

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press. (Cited in pages 84 and 87.)

[Boyd and Vandenberghe, 2007] Boyd, S. and Vandenberghe, L. (2007). Localization and cutting-plane methods. (Cited in pages 85, 87, and 91.)

[Brinker, 2004] Brinker, K. (2004). *Active learning with kernel machines*. PhD thesis, University of Paderborn. (Cited in pages 92, 94, 104, 117, 125, 127, 128, and 131.)

[Bylander, 1994] Bylander, T. (1994). Learning linear threshold functions in the presence of classification noise. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory, COLT 1994, New Brunswick, NJ, USA, July 12-15, 1994.*, pages 340–347. (Cited in pages 4, 38, 39, 40, 49, 133, 136, and 170.)

[Ciresan et al., 2012] Ciresan, D. C., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 3642–3649. (Cited in page 1.)

[Crammer et al., 2006] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585. (Cited in pages 34 and 48.)

[Crammer and Singer, 2003] Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991. (Cited in pages 31, 32, 133, and 139.)

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*. Cambridge University Press. (Cited in page 61.)

[Daniely et al., 2011] Daniely, A., Sabato, S., Ben-David, S., and Shalev-Shwartz, S. (2011). Multiclass learnability and the ERM principle. *Journal of Machine Learning Research - Proceedings Track*, 19:207–232. (Cited in page 71.)

[Dekel et al., 2005] Dekel, O., Shalev-shwartz, S., and Singer, Y. (2005). The forgetron: A kernel-based perceptron on a fixed budget. In *In Advances in Neural Information Processing Systems 18*, pages 259–266. MIT Press. (Cited in page 61.)

[Devroye et al., 1996] Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer. (Cited in pages 19, 144, and 146.)

[Diakonikolas et al., 2011] Diakonikolas, I., O'Donnell, R., Servedio, R. A., and Wu, Y. (2011). Hardness results for agnostically learning low-degree polynomial threshold functions. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1590–1606. (Cited in page 38.)

[Drineas et al., 2006] Drineas, P., Kannan, R., and Mahoney, M. W. (2006). Fast Monte Carlo algorithms for matrices ii: computing a low rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183. (Cited in page 62.)

[Drineas and Mahoney, 2005] Drineas, P. and Mahoney, M. W. (2005). On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(Dec):2153–2175. (Cited in page 62.)

[Dunagan and Vempala, 2008] Dunagan, J. and Vempala, S. (2008). A simple polynomial-time rescaling algorithm for solving linear programs. *Math. Program.*, 114(1):101–114. (Cited in pages 13 and 77.)

[Elbassioni and Tiwary, 2008] Elbassioni, K. M. and Tiwary, H. R. (2008). On computing the vertex centroid of a polyhedron. *CoRR*, abs/0806.3456. (Cited in page 86.)

[Elzinga and Moore, 1975] Elzinga, J. and Moore, T. G. (1975). A central cutting plane algorithm for the convex programming problem. *Math. Program.*, 8(1):134–145. (Cited in page 87.)

[Fanzi et al., 2009] Fanzi, Z., Degui, X., Renfa, L., and Juan, L. (2009). Generalization error bound for the multi-class classification algorithm based on the analytical center of version space [j]. *Journal of Computer Research and Development*, 6:018. (Cited in pages 91 and 115.)

[Fawcett, 2006] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874. (Cited in page 44.)

[Feldman et al., 2012] Feldman, V., Guruswami, V., Raghavendra, P., and Wu, Y. (2012). Agnostic learning of monomials by halfspaces is hard. *SIAM J. Comput.*, 41(6):1558–1590. (Cited in page 38.)

[Floyd and Warmuth, 1995] Floyd, S. and Warmuth, M. K. (1995). Sample compression, learnability, and the vapnik-chervonenkis dimension. *Machine Learning*, 21(3):269–304. (Cited in pages 29 and 30.)

[Franc and Sonnenburg, 2009] Franc, V. and Sonnenburg, S. (2009). Optimized cutting plane algorithm for large-scale risk minimization. *Journal of Machine Learning Research*, 10:2157–2192. (Cited in page 80.)

[Freund and Schapire, 1999] Freund, Y. and Schapire, R. E. (1999). Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, 37(3):277–296. (Cited in page 61.)

[Friess et al., 1998] Friess, T., Cristianini, N., and Campbell, N. (1998). The Kernel-Adatron Algorithm: a Fast and Simple Learning Procedure for Support Vector Machines. In Shavlik, J., editor, *Machine Learning: Proc. of the* $15^{th}$ *Int. Conf.* Morgan Kaufmann Publishers. (Cited in page 61.)

[Garkavi, 1964] Garkavi, A. L. (1964). On chebyshev and almost-chebyshev subspaces. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 28(4):799–818. (Cited in page 87.)

[Gilad-Bachrach and Burges, 2013] Gilad-Bachrach, R. and Burges, C. J. C. (2013). Classifier selection using the predicate depth. *Journal of Machine Learning Research*, 14(1):3591–3618. (Cited in page 94.)

[Golovin and Krause, 2010] Golovin, D. and Krause, A. (2010). Adaptive submodularity: A new approach to active learning and stochastic optimization. *CoRR*, abs/1003.3967. (Cited in page 123.)

[Gonen et al., 2013] Gonen, A., Sabato, S., and Shalev-Shwartz, S. (2013). Efficient active learning of halfspaces: an aggressive approach. *Journal of Machine Learning Research*, 14(1):2583–2615. (Cited in page 123.)

[Graepel et al., 1999] Graepel, T., Herbrich, R., and Obermayer, K. (1999). Bayesian transduction. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 456–462. (Cited in page 104.)

[Graepel et al., 2005] Graepel, T., Herbrich, R., and Shawe-Taylor, J. (2005). Pac-bayesian compression bounds on the prediction error of learning algorithms for classification. *Machine Learning*, 59(1-2):55–76. (Cited in page 30.)

[Grunbaum, 1960] Grunbaum, B. (1960). Partitions of mass-distributions and of convex bodies by hyperplanes. *Pacific Journal of Mathematics*. (Cited in pages 6, 85, 86, 95, 96, 127, 134, 147, 153, and 170.)

[Hfastad, 1997] Hfastad, J. (1997). Some optimal inapproximability results. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 1–10. (Cited in page 38.)

[Haussler, 1988] Haussler, D. (1988). Space efficient learning algorithms. Technical Report UCSC-CRL-88-2, University of Calif. Computer Research Laboratory. (Cited in page 38.)

[Herbrich et al., 2001] Herbrich, R., Graepel, T., and Campbell, C. (2001). Bayes point machines. *Journal of Machine Learning Research*, 1:245–279. (Cited in pages 78, 87, 92, 94, 95, 104, 105, 106, 125, 128, 129, and 170.)

[Höffgen et al., 1995] Höffgen, K., Simon, H., and Horn, K. S. V. (1995). Robust trainability of single neurons. *J. Comput. Syst. Sci.*, 50(1):114–125. (Cited in pages 4 and 38.)

[Joachims et al., 2009] Joachims, T., Finley, T., and Yu, C. J. (2009). Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59. (Cited in pages 99, 107, 108, and 137.)

[Joachims and Yu, 2009] Joachims, T. and Yu, C. J. (2009). Sparse kernel svms via cutting-plane training. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part I*, page 8. (Cited in pages 107, 108, and 114.)

[Johnson and Preparata, 1978] Johnson, D. S. and Preparata, F. P. (1978). The densest hemisphere problem. *Theor. Comput. Sci.*, 6:93–107. (Cited in page 38.)

[Kalai et al., 2008] Kalai, A. T., Klivans, A. R., Mansour, Y., and Servedio, R. A. (2008). Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805. (Cited in page 38.)

[Kannan and Narayanan, 2012] Kannan, R. and Narayanan, H. (2012). Random walks on polytopes and an affine interior point method for linear programming. *Math. Oper. Res.*, 37(1):1–20. (Cited in page 128.)

[Kearns and Li, 1993] Kearns, M. J. and Li, M. (1993). Learning in the presence of malicious errors. *SIAM J. Comput.*, 22(4):807–837. (Cited in page 38.)

[Kearns et al., 1994] Kearns, M. J., Schapire, R. E., and Sellie, L. (1994). Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141. (Cited in page 38.)

[Lampert et al., 2009] Lampert, C. H., Nickisch, H., and Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 951–958. (Cited in page 2.)

[Levin, 1965] Levin, A. (1965). On an algorithm for the minimization of convex functions. *Soviet Math. Doklady*. (Cited in pages 6 and 85.)

[Lewis and Gale, 1994] Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 3–12. (Cited in pages 119 and 125.)

[Li et al., 2005] Li, H., Qi, F., and Wang, S. (2005). A comparison of model selection methods for multi-class support vector machines. In *Computational Science and Its Applications - ICCSA 2005, International Conference, Singapore, May 9-12, 2005, Proceedings, Part IV*, pages 1140–1148. (Cited in pages 31 and 45.)

[Littlestone and Warmuth, 1986] Littlestone, N. and Warmuth, M. K. (1986). Relating data compression and learnability. Technical report. (Cited in pages 29, 30, and 101.)

[Louche and Ralaivola, 2013] Louche, U. and Ralaivola, L. (2013). Unconfused ultraconservative multiclass algorithms. In *JMLR Workshop & Conference Proc. 29, (Proc. of ACML 13)*, pages 309–324. (Cited in pages 6, 52, 135, and 170.)

[Louche and Ralaivola, 2015a] Louche, U. and Ralaivola, L. (2015a). From cutting planes algorithms to compression schemes and active learning. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pages 1–8. (Cited in pages 6, 75, 95, 99, 114, 117, 136, and 170.)

[Louche and Ralaivola, 2015b] Louche, U. and Ralaivola, L. (2015b). Unconfused ultraconservative multiclass algorithms. *Machine Learning*, 99(2):327–351. (Cited in pages 6, 52, 136, and 170.)

[Lovász, 1999] Lovász, L. (1999). Hit-and-run mixes fast. *Math. Program.*, 86(3):443–461. (Cited in page 92.)

[Lovász and Vempala, 2003] Lovász, L. and Vempala, S. (2003). Hit-and-run is fast and fun. (Cited in page 92.)

[Lovász and Vempala, 2006] Lovász, L. and Vempala, S. (2006). Hit-and-run from a corner. *SIAM J. Comput.*, 35(4):985–1005. (Cited in page 128.)

[Machart and Ralaivola, 2012] Machart, P. and Ralaivola, L. (2012). Confusion matrix stability bounds for multiclass classification. *CoRR*, abs/1202.6221. (Cited in page 48.)

[Mercer, 1909] Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446. (Cited in page 26.)

[Minka, 2001a] Minka, T. P. (2001a). Expectation propagation for approximate bayesian inference. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pages 362–369. (Cited in pages 78, 104, 106, and 128.)

[Minka, 2001b] Minka, T. P. (2001b). *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology. (Cited in pages 78, 94, and 106.)

[Mitchell, 1982] Mitchell, T. M. (1982). Generalization as search. *Artif. Intell.*, 18(2):203–226. (Cited in page 104.)

[Moran and Yehudayoff, 2015] Moran, S. and Yehudayoff, A. (2015). Proper PAC learning is compressing. *CoRR*, abs/1503.06960. (Cited in page 29.)

[Morvant et al., 2012] Morvant, E., Koco, S., and Ralaivola, L. (2012). Pac-bayesian generalization bound on confusion matrix for multi-class classification. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. (Cited in page 48.)

[Mount, 2015] Mount, J. (2015). How sure are you that large margin implies low vc dimension? (Cited in page 22.)

[Nesterov, 1995] Nesterov, Y. (1995). Cutting plane algorithms from analytic centers: efficiency estimates. *Mathematical Programming*. (Cited in page 91.)

[Newman, 1965] Newman, D. J. (1965). Location of the maximum on unimodal surfaces. *J. ACM*, 12(3):395–398. (Cited in page 85.)

[Novikoff, 1962] Novikoff, A. (1962). On convergence proofs on perceptrons. *Proceedings of the Symposium on the Mathematical Theory of Automata*, 12:615–622. (Cited in pages 20, 32, 55, and 112.)

[Rademacher, 2007] Rademacher, L. (2007). Approximating the centroid is hard. In *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, pages 302–305. (Cited in page 86.)

[Ralaivola, 2012] Ralaivola, L. (2012). Confusion-based online learning and a passive-aggressive scheme. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 3293–3301. (Cited in pages 48 and 60.)

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408. (Cited in page 20.)

[Rujan, 1997] Rujan, P. (1997). Playing billiards in version space. *Neural Computation*, 9(1):99–122. (Cited in pages 92, 94, 95, 104, 105, 106, and 129.)

[Rujan and Marchand, 1999] Rujan, P. and Marchand, M. (1999). Computing the bayes kernel classifier. (Cited in pages 92 and 105.)

[Schapire and Singer, 2000] Schapire, R. E. and Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168. (Cited in page 34.)

[Schölkopf et al., 2001] Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer theorem. In *Computational Learning Theory, 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, Amsterdam, The Netherlands, July 16-19, 2001, Proceedings*, pages 416–426. (Cited in page 28.)

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels, Support Vector Machines, Regularization, Optimization and Beyond*. MIT University Press. (Cited in page 61.)

[Settles, 2010] Settles, B. (2010). Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11. (Cited in pages 118, 119, and 135.)

[Settles, 2012] Settles, B. (2012). *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. (Cited in page 123.)

[Settles et al., 2007] Settles, B., Craven, M., and Ray, S. (2007). Multiple-instance active learning. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1289–1296. (Cited in page 119.)

[Shashua, 2009] Shashua, A. (2009). Introduction to machine learning: Class notes 67577. *CoRR*, abs/0904.3664. (Cited in pages 19 and 27.)

[Stempfel and Ralaivola, 2007] Stempfel, G. and Ralaivola, L. (2007). Learning kernel perceptron on noisy data and random projections. In *In Proc. of Algorithmic Learning Theory (ALT 07)*. (Cited in page 62.)

[Takerkart and Ralaivola, 2011] Takerkart, S. and Ralaivola, L. (2011). MKPM: a multiclass extension to the kernel projection machine. In *CVPR*, pages 2785–2791. (Cited in page 62.)

[Teo et al., 2010] Teo, C. H., Vishwanathan, S. V. N., Smola, A. J., and Le, Q. V. (2010). Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365. (Cited in page 80.)

[Tong and Koller, 2001] Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66. (Cited in pages 5, 6, 87, 91, 104, 114, 117, 121, 125, 127, 128, 129, 131, 135, and 170.)

[Trapeznikov et al., 2011] Trapeznikov, K., Saligrama, V., and Castañón, D. A. (2011). Active boosted learning (actboost). In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 743–751. (Cited in page 125.)

[Valiant, 1984] Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27:1134–1142. (Cited in page 17.)

[Vapnik, 1998] Vapnik, V. (1998). *Statistical learning theory*. Wiley. (Cited in page 22.)

[Vapnik and Chervonenkis, 1971] Vapnik, V. N. and Chervonenkis, A. Y. (1971). On uniform convergence of the frequencies of events to their probabilities. *Teoriya Veroyatnostei i ee Primeneniya*, 16:264–279. (Cited in page 17.)

[Vert et al., 2004] Vert, J., Tsuda, K., and Schölkopf, B. (2004). *A Primer on Kernel Methods*, pages 35–70. MIT Press. (Cited in page 28.)

[WAHBA, 1990] WAHBA, G. (1990). Spline models for observational data. (Cited in page 28.)

[Watkin, 1993] Watkin, T. (1993). Optimal learning with a neural network. *EPL (Europhysics Letters)*, 21(8):871. (Cited in page 105.)

[Williams and Seeger, 2001] Williams, C. K. I. and Seeger, M. (2001). Using the Nyström Method to Speed Up Kernel Machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press. (Cited in page 62.)

[Zhu, 2005] Zhu, X. (2005). Semi-supervised learning literature survey. (Cited in pages 3 and 118.)

**Titre** Du Bruit de Confusion à l'Apprentissage Actif : Jouer sur la Disponibilité des Étiquettes dans les Problèmes de Classification Linéaire

**Résumé** Les travaux présentés dans cette thèse relèvent de l'étude des méthodes de classification linéaires, c'est à dire l'étude de méthodes ayant pour but la catégorisation de données en différents groupes à partir d'un jeu d'exemples, préalablement étiquettés, disponible en amont et appelés ensemble d'apprentissage. En pratique, l'acquisition d'un tel ensemble d'apprentissage peut être difficile et/ou couteux, la catégorisation d'un exemple étant de fait plus hardu que l'obtention de du-dis exemple. Cette disparité entre la disponibilité des données et notre capacité à constituer un ensemble d'apprentissage étiqueté a été un des problèmes centraux de l'apprentissage automatique et ce manuscrit s'interèsse à deux solutions usuellement considérées pour contourner ce problème : l'apprentissage en présence de données bruitées et l'apprentissage actif. Plus précisement, nous nous intéressons en premier lieu à l'apprentissage avec bruit de confusion (ou, alternativement, apprentissage confus) où chaque exemple, en fonction de sa catégorie d'origine uniquement, a une probabilité fixe d'être incorrectement étiqueté dans l'ensemble d'apprentissage. Alternativement, l'apprentissage actif se focalise sur l'utilisation d'un très petit nombre d'exemples en laissant au processus d'apprentissage la liberté de demander de manière autonome les catégories des exemples jugés importants. D'une manière plus générale, ces deux approches peuvent être vue comme jouant sur la disponibilité des étiquettes, soit en acceptant un étiquetage de moins bonne qualité et potentiellement faux, soit en limitant le nombre d'étiquettes nécéssaires à l'apprentissage à son strict minimum. Notre première contribution est de proposer un algorithme d'apprentissage pour le cadre de l'apprentissage confus multiclasse là où les précédents travaux à ce sujet ne s'étaient portés que sur le cas des problèmes avec seulement deux classes [Blum et al., 1998, Bylander, 1994]. Notre étude démontre que l'algorithme que nous proposons est capable d'inférer une fonction de classification correcte à partir d'un ensemble d'apprentissage confus, à supposer que la matrice de confusion associée au bruit est fournie. Notre travail sur ce sujet à été l'occasion de deux publications, à l'*Asian Conference on Machine Learning* en 2013 [Louche and Ralaivola, 2013], et dans le journal *Machine Learning* en version étendue [Louche and Ralaivola, 2015b]. Notre seconde contribution est de proposer une nouvelle approche d'apprentissage actif inspirée des travaux de [Tong and Koller, 2001] et plus particulièrement des similarités entre l'algorithme SIMPLE et les méthodes de Plans Coupants. Plus particulièrement, nous nous proposons de revisiter les principes de l'apprentissage actif et par extension, de l'apprentissage supervisé en général, au travers d'une approche plus géométrique. Notre contribution à l'apprentissage actif est un nouvel algorithme, que nous appelons ACTIVE-BPM en référence à l'algorithme des Machines à Point de Bayes [Herbrich et al., 2001], inspiré par la théorie que nous developpons et basé sur les centres de gravité. Une particularité intéréssante de notre algorithme est nottament sa robustesse aux problèmes dont les espaces de version sont connus pour poser problème à l'algorithme du SIMPLE. Nous proposons également deux contributions auxiliaires à ce résultat qui sont le fruit de notre étude sur les problèmes d'apprentissage actif. Le premier est un nouvel algorithme d'apprentissage supervisé interfaçant les méthodes de Plans Coupants avec l'algorithme de Perceptron. Notre dernière contribution est un théorème général de géometrie sur les centres de gravité approximés. Plus précisément, ce dernier résultat porte sur l'applicabilité du théorème de [Grunbaum, 1960] concernant la partition des corps convexes par leur centre de gravité lorsque à la place du vrai centre de gravité, seule une approximation est disponible. Ces trois contributions ont précédemment étaient publiées ensemble lors de l'*International Joint Conference on Neural Network* en 2015 [Louche and Ralaivola, 2015a]. De plus, cette publication a été récompensée du prix du meilleur papier étudiant lors de sa présentation en conférence.

**Mots-clés** modèles linéaires, classification, multi-classe, matrice de confusion, bruit, apprentissage actif, géométrie computationelle, perceptrons, méthodes de plans coupants, schémas de compréssion

**Title** From Confusion Noise to Active Learning: Playing on Label Availability in Linear Classification Problems

**Abstract** The works presented in this thesis fall within the general framework of linear classification, that is the problem of categorizing data into two or more classes based on on a training set of labelled data. In practice though acquiring labeled examples might prove challenging and/or costly as data are inherently easier to obtain than to label. Dealing with label scarceness have been a motivational goal in the machine learning literature and this work discuss two settings related to this problem: learning in the presence of noise and active learning. More precisely, the first setting we consider is the one of learning under confusion noise (that we may refer to as confused learning for short) where the probability for an example to have a corrupted label only depends of its true class. Conversely, in Active Learning the emphasis is put on using as few labels as possible by allowing for interaction between the learning algorithm and the labelling process. More generally both of these approaches can be thought as playing on label availability whether it is by accepting erroneous, albeit easy to get, data or, at the contrary, by trying to reduce the label needs of the learning algorithm to its optimal minimum. Our first contribution is to propose and study an algorithm that is able to learn in a multiclass confused setting whereas previous works on this subject [Blum et al., 1998, Bylander, 1994] have been limited to the case of bi-class classification. Notably our algorithm is proved to converge to the true, unknown, labelling function given it has access to the confusion matrix associated with the noising process. Our work on this topic have previously been the subject of two publications, first at the *Asian Conference on Machine Learning* in 2013 [Louche and Ralaivola, 2013] and then as an extended revision in *Machine Learning Journal* [Louche and Ralaivola, 2015b]. Our second contribution is to propose a novel Active learning algorithm inspired from the similarities between the SIMPLE algorithm [Tong and Koller, 2001] and Cutting Planes methods. We revisit the setting of active learning, and more generally supervised learning, through a geometric point of view. Concretely, our contribution take the form of a theoretically backed algorithm for active learning based on centers of gravity that we call ACTIVE-BPMin reference to the Bayes Point Machine algorithm [Herbrich et al., 2001]; notably theoretical and practical results show that our algorithm does not suffer from the pitfalls of SIMPLE on ill-shaped version spaces. We also propose two auxiliary contributions that come as a by-product of the above study. The first one is a new supervised algorithm interfacing a Cutting Planes update scheme with a perceptron algorithm. Our last contribution comes in the form of a general theorem on approximate centers of gravity. More precisely, we study how the result of [Grunbaum, 1960] on the partition of convex bodies by centers of gravity degrades when approximations are used. Those three last contributions have been first published together at the *International Joint Conference on Neural Network* in 2015 [Louche and Ralaivola, 2015a] and won the Best Student Paper award during the conference meeting.

**Keywords** linear models, classification, multiclass, confusion matrix, noise, active learning, computational geometry, perceptron methods, cutting planes methods, compression schemes